

4

INTRODUCCIÓN AL CSS3

El lenguaje CSS (Cascading Style Sheets) es un lenguaje de marcado de presentación dedicado a la elaboración de páginas web. Fue definido por primera vez en 1994 y, poco después, fue añadido al grupo de trabajo de la W3C como parte del proceso de desarrollo y estandarización.

El crecimiento de especificación CSS ha sido francamente irregular con respecto a otras evoluciones de lenguajes de marcado. De hecho, sólo hay que investigar un poco para ver que en 1998 se publicó la recomendación 2.1 y, poco después, en 1999, ya apareció el primer borrador de la versión actual de CSS. Desde entonces, ha ido evolucionando hasta la especificación que es la actualmente denominamos CSS3.

El lenguaje de marcado CSS3 está pensado para separar el contenido y estructura de la presentación en un documento HTML, XML o XHTML. Esta separación permite, además, mejorar la accesibilidad y usabilidad de los documentos porque proporciona una mayor flexibilidad, una mejor reutilización de código y una reducción de la complejidad gracias, entre otras cosas, a que se evita la repetición de código.

Entre sus características iniciales podemos destacar que es un lenguaje sencillo que permite diferentes métodos de renderizado y que todo se realiza a través de reglas que se aplican en función de unos selectores previamente definidos.

4.1 SOPORTE A LOS NAVEGADORES

En líneas generales podemos afirmar que Chrome, Edge, Firefox, Internet Explorer, Opera y Safari son los navegadores más utilizados por los usuarios y que, prácticamente todos, salvo algunas excepciones, dan soporte a la inmensa mayoría de sus propiedades, selectores, pseudo-elementos y pseudo-clases.

En la dirección https://www.w3schools.com/cssref/css3_browsersupport.asp es posible encontrar el soporte específico para todas las propiedades CSS de manera individual.

4.2 /CÓMO FUNCIONA CSS

Básicamente, el lenguaje CSS es una especificación que define un conjunto de reglas, cada una de ellas, definida a través de un selector y una declaración.

Selector	Declaración
div	{ background: white; font-family: Arial; font-size: 14px }

↑ ↑ ↑ ↑ ↑ ↑
Propiedad Valor Propiedad Valor Propiedad Valor

Como se puede apreciar en la ilustración anterior, la regla CSS está compuesta por un selector y un conjunto de declaraciones, en este caso dos.

El selector, puede ser un nombre de etiqueta HTML, una clase, un identificador, un comodín o, incluso, una combinación de ellos. Su objetivo es localizar el elemento o grupo de elementos dónde se debe aplicar este conjunto de declaraciones. La declaración, por su parte, está definida a modo de bloque mediante unas llaves y representa los estilos a aplicar a través de pares de propiedad-valor.

Cada uno de estos pares siempre termina con un punto y coma, aunque si es el final de la declaración puede omitirse, como es el caso de la ilustración. El conjunto de estos pares de propiedad-valor es el que definirá el comportamiento visual del objeto que se haya seleccionado por el selector previamente definido.

4.3 DEFINICIÓN DE SELECTOR Y CLASIFICACIÓN

Como se acaba de comentar, un selector, puede ser un nombre de etiqueta HTML, una clase, un identificador, un comodín o, incluso, una combinación de ellos y se utilizan para localizar o seleccionar el conjunto de elementos HTML que se desea manipular.

Aunque los selectores pueden ser de muy diversos tipos, se puede establecer una clasificación general en la que se dividen en universales, simples, combinados, de atributo, pseudo-clases y pseudo-elementos. A continuación, se describen todos y cada uno de ellos brevemente ya que, posteriormente, se verán en detalle:

Tipo	Descripción
Simples	Aquellos que permiten seleccionar los elementos a partir del nombre de una etiqueta, identificador o clase. Ejemplos de selectores simples son “ button ”, “ #banner ” o “ .header ”.
Combinados	Aquellos que permiten seleccionar los elementos a partir de una relación preestablecida entre ellos. Ejemplos de selectores combinados son “ div p ”, “ #banner .title ” o “ .header .title ”.
Pseudo-clases	Son un tipo especial de selectores que permiten seleccionar los elementos a partir del estado en el que se encuentran y siempre van precedidos del símbolo dos puntos. Ejemplos de este tipo de selectores son “ button:hover ”, “ a:active ” o “ input:focus ”.
Pseudo-elementos	Son otro tipo especial de selectores que permiten seleccionar partes concretas de los elementos o del documento. Siempre van precedidos del símbolo de dos puntos, repetido dos veces. Ejemplos de este tipo de selectores son “ ::selection ”, “ .icon:focus ” o “ input::placeholder ”.
De atributo	Aquellos que permiten seleccionar los elementos a partir de sus atributos o propiedades y siempre van declarados a través de corchetes. Algunos ejemplos podrían ser “ a[href^=”https”] ”, “ a[href*=”blog”] ” o “ a[href\$=”php”] ”.
Universal	Es un selector que permite la selección de todos los elementos de un elemento dado, incluyendo el propio documento. Ejemplos de uso de este selector son “ * ” o “ article * ”.

4.4 UNIDADES DE MEDIDA

Existen dos tipos de unidad de medida, las absolutas y las relativas.

4.4.1 Unidades absolutas o fijas

Las unidades absolutas o fijas son aquellas que no están relacionadas de ninguna forma con el contexto ni ninguna otra entidad. Entre las diferentes propuestas que ofrece CSS como unidades de medida absolutas podemos encontrar:

Unidad	Descripción
px	El término PX hace referencia a los puntos de la pantalla, por lo que cada punto en la pantalla es un pixel. Sus posibles valores pueden ser negativos y positivos, dependiendo de la propiedad y contexto.
pt	El término PT hace referencia al tamaño del punto en una pantalla o papel. En lo referente a su equivalencia, un punto se suele traducir a 0.35mm, o lo que es lo mismo, a 1.33 píxeles. Sus posibles valores pueden ser negativos y positivos, dependiendo de la propiedad y contexto.
in	El término IN hace referencia a la longitud del sistema inglés pulgada, es decir, 2.54cm. En lo referente a sus posibles equivalencias, una pulgada se suele traducir en a 96 píxeles o, lo que es lo mismo, 72 puntos. Sus posibles valores pueden ser negativos y positivos, dependiendo de la propiedad y contexto.
cm, mm, pc	Los términos CM (centímetros), MM (milímetros) y PC (picas) hacen referencia las conocidas medidas de longitud. Hoy en día es poco frecuente si uso y, en lo referente a su equivalencia, podríamos decir que una pica es equivalente a 12 puntos, o lo que es lo mismo, 4.23 milímetros.

4.4.2 Unidades relativas

Las unidades relativas son aquellas que toman como punto de referencia otra entidad o contexto. Entre las diferentes propuestas que ofrece CSS como unidades de medida absolutas tenemos:

Unidad	Descripción
%	El término % hace referencia al porcentaje del marco o contexto actual. Sus valores no tienen por qué ir de cero a cien, ya que también son válidos los valores superiores.
ch	El término CH hace referencia a la anchura, en píxeles, del carácter 0 del elemento padre, esto es, depende del tamaño de la fuente y letra base realizar la equivalencia. El tamaño que se toma como referencia es la anchura del carácter 0, por tanto, si se establece el ancho de un elemento a 80ch, lo que se está indicando es que, será de ancho como 80 caracteres 0, todos seguidos sin espacios en blanco.
em	El término EM hace referencia a la anchura, en píxeles, de la letra M mayúscula del elemento padre, esto es, si el elemento padre tiene aplicado un tamaño de fuente de 14 píxeles, 1.2em será equivalente a 16.8 píxeles.

ex	El término EX hace referencia a la anchura, en píxeles, de la letra X minúscula del elemento padre, esto es, depende del tamaño de la fuente y letra base realizar la equivalencia. El tamaño que se toma como referencia es la anchura de la X minúscula, por tanto, si se establece el ancho de un elemento a 80EX, lo que se está indicando es que, será de ancho como 80 caracteres x, todos seguidos sin espacios en blanco.
rem	El término REM hace referencia a la anchura, en píxeles, de la letra M mayúscula del elemento HTML (o raíz), esto es, si el elemento HTML tiene aplicado un tamaño de fuente de 14 píxeles, 1.2rem será equivalente a 16.8 píxeles.
vmax	El término VMAX hace referencia a la centésima parte de la altura o anchura de la ventana gráfica o área útil. La regla de para saber qué valor se está aplicando es “cuál de los valores de ancho y alto es mayor”, esto es, si la altura es mayor que la anchura, 1VMAX será equivalente a 1VH, pero si la anchura es mayor que la altura, 1VMAX será equivalente a 1VW.
vmin	El término VMIN hace referencia a la centésima parte de la altura o anchura de la ventana gráfica o área útil. La regla de para saber qué valor se está aplicando es “cuál de los valores de ancho y alto es menor”, esto es, si la altura es menor que la anchura, 1VMIN será equivalente a 1VH, pero si la anchura es menor que la altura, 1VMIN será equivalente a 1VW.
vh	El término VH hace referencia a la centésima parte de la altura del viewport, esto es, un 1% de la altura de la ventana gráfica o área útil. Por tanto, si se establece el alto de un elemento a 100VH, lo que se está indicando es que sea el 100% del alto del viewport, o lo que es lo mismo, 100% del alto de la ventana del navegador.
vw	El término VW hace referencia a la centésima parte de la anchura del viewport, esto es, un 1% de la anchura de la ventana gráfica o área útil. Por tanto, si se establece el ancho de un elemento a 100VW, lo que se está indicando es que sea el 100% del ancho del viewport, o lo que es lo mismo, 100% del ancho de la ventana del navegador.

4.5 SELECTORES

Como ya uno se podrá haber dado cuenta, un selector es una entidad que permite enlazar o vincular unos estilos con los elementos que conforman un documento.

A continuación, se muestra una descripción, más o menos detallada, con los selectores CSS más frecuentemente utilizados en documentos y páginas web.

4.5.1 Simples y combinados

Selector	Descripción
*	El carácter “asterisco” es un selector universal que selecciona todos los elementos del documento. <code>* { color: black; }</code>
Tipo o etiqueta	Permite seleccionar todos los elementos que se correspondan con un nombre de etiqueta HTML. <code>div { border-color: #00FF00; }</code>
.	El carácter “punto” selecciona todos los elementos que tengan como clase el identificador indicado detrás del punto. <code>.items-group { color: #000; }</code>
#	El símbolo “almohadilla” permite seleccionar el elemento que tenga como ID el identificador indicado detrás de la almohadilla. Aunque no provoca error que haya dos elementos con el mismo ID, es importante controlarlo para no obtener resultados imprevisibles. <code>#banner { background: black; color: white; }</code>
Espacio	El carácter “espacio” permite seleccionar todos los elementos que estén contenidos en los elementos que se correspondan con el selector previamente declarado. <code>div p { font-size: 14px; }</code>
,	El carácter “coma” permite seleccionar todos los elementos que resulten de combinar los selectores de forma independiente. <code>div, span { color: red; }</code>
>	El símbolo “mayor que” permite seleccionar todos los elementos que estén contenidos en los elementos que se correspondan con el selector previamente declarado y que sean hijos directos. <code>div > p { width: 100%; }</code>
+	El símbolo “suma” permite seleccionar todos los elementos que resulten del selector de la derecha y que estén declarados justo después del de los elementos seleccionados por el selector predecesor. <code>input + button { border: 2px solid gray; }</code>
~	La virgulilla permite seleccionar todos los elementos que sean hermanos del elemento previamente declarado. <code>div ~ label { border-color: #00FF00; }</code>

[atributo]	Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes. <pre>span[role] { width: 100%; }</pre>
[atributo=valor]	Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor sea el “valor” indicado. <pre>input[type="image"] { height: 32px; }</pre>
[atributo^=valor]	Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor empiece por el “valor” indicado. Esto seleccionaría, por ejemplo, todos los elementos que tengan el atributo CLASS establecido a FORM-GROUP, FORM-CONTROL, pero no ACTION-FORM. <pre>input[class^="form"] { height: 32px; }</pre>
[atributo*=valor]	Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor contengan el “valor” indicado. Esto seleccionaría, por ejemplo, todos los elementos que tengan el atributo CLASS establecido a FORM-GROUP, FORM-CONTROL y ACTION-FORM. <pre>input[class*="form"] { height: 32px; }</pre>
[atributo\$=valor]	Permite seleccionar los elementos que tengan definido el atributo establecido entre corchetes y cuyo valor termine por el “valor” indicado. Esto seleccionaría, por ejemplo, todos los elementos que tengan el atributo CLASS establecido a ACTION-FORM, pero no FORM-CONTROL y FORM-GROUP. <pre>input[class\$="form"] { height: 32px; }</pre>

4.5.2 Pseudo-clases

Las pseudo-clases son un tipo de selector que tiene CSS y que sirven para dar funcionalidad a los elementos en función de su estado. A continuación, se explican la mayoría de ellas.

Pseudo-clase	Descripción
:active	Permite seleccionar todos los elementos de tipo enlace que estén activos. <pre>a:active { color: #003366; }</pre>
:checked	Permite seleccionar todos los elementos de tipo RADIO, CHECKBOX u OPTION que estén chequeados o commutados, es decir, que los elementos de tipo RADIO, CHECKBOX tengan establecida la propiedad CHECKED o que los elementos OPTION de un elemento SELECT tengan establecida la propiedad SELECTED. <pre>input[type="radio"]:checked { background: #000; }</pre>

:disabled	Permite seleccionar todos los elementos que estén deshabilitados, es decir, que tengan establecida la propiedad DISABLED. <code>input:disabled{ background: #eee; color: #999; }</code>
:empty	Permite seleccionar todos los elementos que estén totalmente vacíos, es decir, que no tengan ni espacios, ni texto, ni hijos. <code>span:empty{ display: none; }</code>
:enable	Permite seleccionar todos los elementos que estén habilitados, es decir, que no tengan establecida la propiedad DISABLED. <code>input:enabled{ border: 1px solid #fff; }</code>
:first-child	Permite seleccionar los elementos que sean el primer descendiente del elemento especificado. Si no indica elemento, se seleccionan todos los primeros descendientes. <code>ul li:first-child { font-weight: bold; }</code>
:first-of-type	Permite seleccionar los elementos que sean el primer descendiente del elemento y tipo (nombre de etiqueta) especificado. Si no indica elemento, se seleccionan todos los primeros descendientes. La diferencia con :FIRST-CHILD es que, este, seleccionará los elementos por tipo y no por posición. <code>div span:first-of-type { font-weight: bold; }</code>
:focus	Permite seleccionar el elemento que posea el foco de teclado. <code>a:focus { background: #003366; color: #FFFFFF; }</code>
:hover	Permite seleccionar todos los elementos que estén por dentro del elemento por el que se pasa el puntero del ratón. <code>a:hover { background: #003366; color: #FFFFFF; }</code>
:invalid	Permite seleccionar todos los elementos que no pasen la validación requerida por HTML5. El selector :INVALID sólo funciona cuando los elementos tienen alguna limitación como que sea requerido, que tenga un mínimo o máximo establecido o que sea de un campo de tipo especial como el email o number. Por tanto, si un elemento INPUT tiene el atributo REQUIRED establecido y se define una regla que contemple esta pseudo-clase, se aplicará de manera automática. <code>input:invalid { background: red; }</code>
:last-child	Permite seleccionar los elementos que sean el último descendiente del elemento especificado. Si no indica elemento, se seleccionan todos los últimos descendientes. <code>ul li:last-child { font-weight: bold; }</code>

:last-of-type	Permite seleccionar los elementos que sean el último descendiente del elemento y tipo (nombre de etiqueta) especificado. Si no indica elemento, se seleccionan todos los últimos descendientes. La diferencia con :LAST-CHILD es que, este, seleccionará los elementos por tipo y no por posición. <pre>div p:last-of-type { font-weight: bold; }</pre>
:link	Permite seleccionar todos los elementos de tipo enlace no visitados. <pre>a:link { color: red; text-decoration: none; }</pre>
:optional	Permite seleccionar todos los elementos de tipo INPUT, SELECT y TEXTAREA que sean opcionales, es decir, que no tengan establecido la propiedad REQUIRED. <pre>input:optional { background: gold; color: #000; }</pre>
:only-child	Permite seleccionar los elementos que sean el único descendiente entre los elementos del mismo nivel. <pre>ul li:only-child { font-weight: bold; }</pre>
:only-of-type	Permite seleccionar los elementos que sean el único descendiente de ese tipo o nombre de etiqueta entre los elementos del mismo nivel. <pre>div span:only-of-type { font-weight: bold; }</pre>
:not	Permite seleccionar todos los elementos que NO coincidan con el selector indicado entre paréntesis. El valor dentro de los paréntesis puede ser cualquier selector de los anteriores, es decir, puede ser una etiqueta, una clase, un identificador, una pseudo-clase, un pseudo-elemento o una combinación de ellos. <pre>*:not(a) { color: black; }</pre>
:nth-child(n), :nth-of-type	Permite seleccionar los elementos que sean el número de descendiente indicado por el valor de N, empezando por 1. Cuando se trabaja con este selector, se pueden utilizar valores de palabra clave (también llamados valores predefinidos) y de notación funcional (también llamados valores calculados). Los valores de palabra clave son las palabras reservadas EVEN y ODD, que representan a los valores pares e impares respectivamente. Los valores de notación funcional utilizan el formato de expresión AN + B, que representa la serie de valores que resultan de multiplicar un factor N por A y sumarle el valor de B. Dicho de otro modo, el valor del patrón AN + B representa a los elementos que resulten de realizar la división entera de A más el offset entero B. Por tanto, el patrón 2N simboliza todos los elementos pares y, 3N+1, simboliza los elementos que estén definidos en las posiciones 1, 4, 7, ... La diferencia entre :NTH-CHILD y :NTH-OF-TYPE es que, este último, seleccionará los elementos por tipo y no por posición. <pre>li:nth-child(3) { font-weight: bold; }</pre>

:nth-last-child(n)	<p>Permite seleccionar los elementos que sean el número de descendiente indicado por el valor de N empezando desde el final y con el valor 1. A modo aclaratorio, el valor de N igual a 1 se corresponderá con el último descendiente del elemento padre, el valor de N igual a 2, se corresponderá con el penúltimo descendiente, y así sucesivamente. Cuando se trabaja con este selector, se pueden utilizar valores de palabra clave (también llamados valores predefinidos) y de notación funcional (también llamados valores calculados). Los valores de palabra clave son las palabras reservadas EVEN y ODD, que representan a los valores pares e impares respectivamente.</p> <p>Los valores de notación funcional utilizan el formato de expresión AN + B, que representa la serie de valores que resultan de multiplicar un factor N por A y sumarle el valor de B. Dicho de otro modo, el valor del patrón AN + B representa a los elementos que resulten de realizar la división entera de A más el offset entero B. Por tanto, el patrón 2N simboliza todos los elementos pares y, 3N+1, simboliza los elementos que estén definidos en las posiciones 1, 4, 7, ...</p> <pre>li:nth-last-child(2) { font-weight: bold; }</pre>
:nth-last-of-type(n)	<p>Selecciona los elementos que sean del tipo especificado y cuya posición sea el valor indicado por N empezando desde el final, con el valor 1. A modo aclaratorio, el valor de N igual a 1 se corresponderá con el último descendiente del elemento padre que tenga ese tipo (nombre de etiqueta), que no tiene por qué ser el último descendiente.</p> <p>Cuando se trabaja con este selector, se pueden utilizar valores de palabra clave (también llamados valores predefinidos) y de notación funcional (también llamados valores calculados). Los valores de palabra clave son las palabras reservadas EVEN y ODD, que representan a los valores pares e impares respectivamente.</p> <p>Los valores de notación funcional utilizan el formato de expresión AN + B, que representa la serie de valores que resultan de multiplicar un factor N por A y sumarle el valor de B. Dicho de otro modo, el valor del patrón AN + B representa a los elementos que resulten de realizar la división entera de A más el offset entero B. Por tanto, el patrón 2N simboliza todos los elementos pares y, 3N+1, simboliza los elementos que estén definidos en las posiciones 1, 4, 7, ...</p> <pre>div p:nth-last-of-type(2) { font-weight: bold; }</pre>
:read-only	<p>Permite seleccionar todos los elementos que tengan el atributo READONLY establecido entre sus atributos HTML, es decir, que sean de sólo lectura.</p> <pre>span:read-only { background: lime; }</pre>
:required	<p>Permite seleccionar todos los elementos que tengan el atributo REQUIRED establecido entre sus atributos HTML, es decir, que tengan un carácter obligatorio.</p> <pre>span:invalid:required { background: orange; }</pre>

:root	Permite seleccionar el elemento raíz del documento, al igual que el elemento HTML, pero con la diferencia de que, la pseudo-clase :ROOT tiene mucho más peso y relevancia. Por ejemplo, mientras que el selector HTML no se suele utilizar para definir variables CSS, la pseudo-clase :ROOT sí. <pre>:root { --bg-color: #F00; }</pre>
:target	Las direcciones o URL que comienzan con una almohadilla representan un identificador de anclaje a un determinado elemento dentro de un documento. Cuando el usuario pulsa en un elemento A que posee un HREF con una almohadilla, se lanza un evento que pone el foco en el elemento que tiene ese ID. Cuando eso sucede se activa la pseudo-clase :TARGET. Por tanto, la pseudo-clase :TARGET selecciona el elemento destino cuando el usuario pulsa en un elemento que vincula dicho elemento con el punto de anclaje. <pre><style> div { display: none } </style> Enlace <div id="enlace"> <h2>Enlace</h2> <p>Un contenido cualquiera</p> </div></pre> <p>Si ejecutásemos el código anterior en un navegador, lo que se vería es sólo el enlace y si pulsásemos en dicho enlace, no mostraría nada, únicamente provocaría una navegación hacia el punto de anclaje. Para hacerlo visible cuando se pulse el enlace podríamos recurrir a una funcionalidad en JavaScript, pero no, es más sencillo a través de la pseudo-clase :TARGET.</p> <pre>div:target { display: block; }</pre> <p>Ahora sí, si añadimos esta regla al código anterior, y pulsamos en el enlace, veremos que, además de provocar una navegación, se muestra el DIV con todo su contenido.</p>
:valid	Permite seleccionar todos los elementos que pasen la validación requerida por HTML5 y JavaScript. El selector :VALID sólo funciona cuando los elementos tienen alguna limitación como que sea requerido, que tenga un mínimo o máximo establecido, que sea de un campo de tipo especial como el email o sólo numérico. Por tanto, si un elemento INPUT no tiene el atributo REQUIRED establecido, no realiza ningún otro tipo de validación, y se define una regla que contempla esta pseudo-clase, se aplicará de manera automática.
:visited	Permite seleccionar todos los elementos de tipo enlace visitados. <pre>a:visited { color: green; }</pre>

4.5.3 Pseudo-elementos

Los pseudo-elementos son un tipo de selector que tiene CSS y que sirven para añadir estilos adicionales en una parte concreta del elemento. Es por ello que se suele afirmar que, mientras que las pseudo-clases están vinculadas a los estados del elemento, los pseudo-elementos están vinculados al aspecto visual.

Pseudo-elemento	Descripción
<code>::before,</code> <code>::after</code>	<p>Prácticamente todos los elementos de CSS tienen tres capas de visualización superpuestas, la anterior, la seleccionada y la posterior. Mientras que la capa seleccionada es manipulable a través del propio elemento, la anterior y posterior son manipulables a través de los pseudo-elementos BEFORE y AFTER.</p> <p>Se dice que los pseudo-elementos BEFORE y AFTER son el primer y último hijo del elemento, respectivamente, sin embargo, aunque sean hijos del elemento, no pertenecen al HTML. Por esta razón, cuando se desea insertar contenido en estas capas se debe recurrir a la propiedad CONTENT, para asignar contenido sin la intervención de HTML.</p> <p>Aunque pueda parecer que no, el contenido de la propiedad CONTENT puede ser muy diverso. Permite la asignación de una cadena de texto (vacía o no), un código hexadecimal o Unicode, una URL de una imagen o archivo, una palabra clave de CSS como OPEN-QUOTE o CLOSE-QUOTE, una función CSS como ATTR o, incluso, una combinación de todos ellos.</p> <pre>.elemento::before { content: " "; } button i::after { content: url(./icono.png); } h2 span::before { content: attr(data-value); } div::after{content: open-quote " X " close-quote}</pre>
<code>::selection</code>	<p>Permite definir un estilo específico para marcar el texto seleccionado. Es importante destacar que este pseudo-elemento sólo puede utilizar las propiedades BACKGROUND, COLOR, CURSOR y OUTLINE.</p> <pre>::selection { background: black; color: yellow; }</pre>
<code>::placeholder</code>	<p>Permite definir un estilo específico para los elementos que admiten el establecimiento del atributo PLACEHOLDER de HTML.</p> <pre>::placeholder { background: whitesmoke; }</pre>

4.5.4 Especificidad de los selectores

La especificidad es la forma mediante la cual los agentes de usuario deciden que tiene más o menos relevancia o peso. Esto es una de las características que hará que una regla se aplique y sobrescriba a otras, independientemente de donde esté declarada, antes o después.

Por resumirlo en una frase, los selectores de **tipo** y **pseudo-elementos** poseen un peso de 0, los selectores de **clase** y **atributo** (con corchetes) poseen un peso de 1, los

selectores de **ID** (con almohadilla) poseen un peso de 2, el atributo **style** posee un peso de 3, la palabra clave **!important** posee un peso de 4 y el **atributo style con !important** dentro posee un peso de 5.

Tanto el selector universal, como los combinadores (+, >, ~ o el espacio) y la pseudo-clase :**NOT()**, no tienen efectividad en lo que respecta a estos pesos, pero sí dentro de la definición de la pseudo-clase :**NOT()**.

Cualquier declaración en línea sobre los elementos HTML sobrescribe los estilos definidos en las hojas de estilo internas o externas, a menos que, las reglas CSS tengan contenidas la palabra clave **IMPORTANT**.

Aunque, en principio, esta palabra clave no tiene nada que ver con la especificidad, sí que influye en ella anulando prácticamente toda posible herencia. Por esta razón, se ha puesto con el mayor peso en nuestra tabla.

También cabe destacar que, el uso de la palabra clave **IMPORTANT** no es una buena práctica a nivel de desarrollo profesional, primero porque rompe la estandarización de las reglas en cascada, segundo, porque dificulta la lectura y depuración del código y, tercero, pero no menos importante, porque puede provocar comportamientos no contemplados o deseados en el documento.

4.6 PROPIEDADES

A continuación, se muestra una lista con las propiedades de CSS3 más frecuentes en el diseño de páginas web agrupadas por funcionalidad.

4.6.1 Texto, fuentes y tipos de letra

4.6.1.1 PROPIEDAD FONT-FAMILY

Especifica el espacio entre caracteres. Entre sus posibles valores podemos encontrar **NORMAL**, que indica el valor prefijado por el agente de usuario y **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS.

Si se desea indicar varias fuentes, se puede hacer a través del carácter coma, no obstante, su aplicación será en función de la disponibilidad y de izquierda a derecha, es decir, si la primera no está disponible o no se puede seleccionar, se seguirá intentando con las siguientes.

4.6.1.2 PROPIEDAD FONT-SIZE

Especifica el tamaño del texto. Entre sus posibles valores podemos encontrar valores desde **XX-SMALL**, que indica un tamaño de fuente baladí, equivalente a 9px, hasta **XX-LARGE**, que indica un tamaño de fuente enorme, equivalente a 32px. No obstante, también es posible indicar un valor establecido en una de las medidas permitidas de CSS que se comentaron anteriormente.

4.6.1.3 PROPIEDAD FONT-STYLE

Especifica el estilo del texto. Entre sus posibles valores podemos encontrar **NORMAL**, que indica que se muestre con un estilo de fuente normal, **ITALIC**, que indica que se muestre con un estilo de letra cursiva y, **OBLIQUE**, que indica que se muestre con un estilo de fuente oblicuo.

4.6.1.4 PROPIEDAD FONT-VARIANT

Especifica la variación del texto. Esta propiedad presenta múltiples valores, pero el único que, de verdad se utiliza, es **SMALL-CAPS** y **ALL-SMALL-CAPS** para indicar que el texto se muestre con las letras minúsculas convertidas a mayúsculas, pero con un tamaño menor.

4.6.1.5 PROPIEDAD FONT-WEIGHT

Especifica el grosor del texto.

Entre sus posibles valores podemos encontrar **NORMAL**, que es el valor por defecto e indica que se muestre la letra con un grosor normal, **BOLD**, que indica que se muestren los caracteres gruesos y, **100, 200, 300, 400, 500, 600, 700, 800** y **900**, que indican que se muestre el grosor de los caracteres en función de estos valores.

Muchas fuentes vectoriales utilizan esta propiedad para definir diferentes pesos o grosores de carácter. En general, el valor **100** se corresponde con un tipo de letra muy fina o delgada, el valor **400** a un tipo **NORMAL** y **700** o superior a un tipo **BOLD**.

4.6.1.6 PROPIEDAD LETTER-SPACING

Especifica el espacio entre caracteres que se debe aplicar al texto. Entre sus posibles valores podemos encontrar **NORMAL**, que indica que se muestre la letra con un espacio entre caracteres normal y es el valor por defecto. No obstante, también es posible indicar un valor establecido en una de las medidas permitidas de CSS que se comentaron anteriormente.

4.6.1.7 PROPIEDAD LINE-HEIGHT

Especifica el espacio entre líneas de un texto. Entre sus posibles valores podemos encontrar **NORMAL**, que indica que se muestre la letra con un espacio entre caracteres normal definido por defecto, un **[NÚMERO]**, que indica un factor de multiplicación para el tamaño de fuente actual y, como no, un **[VALOR]** que indica un valor establecido en una de las medidas permitidas de CSS comentadas anteriormente.

4.6.1.8 PROPIEDAD TEXT-ALIGN

Especifica la alineación del texto. Entre sus posibles valores podemos encontrar **LEFT**, **RIGHT**, **CENTER** y **JUSTIFY** que indican que el texto se alinee a la izquierda, a la derecha, de manera centrada o de forma justificada, respectivamente.

NOTA

Cabe destacar que la alineación justificada no suele ser una buena opción si se desea que la página sea accesible.

4.6.1.9 PROPIEDAD TEXT-DECORATION

Especifica la decoración agregada al texto. Aunque esta propiedad presenta múltiples valores, los más recurrentes y frecuentemente utilizados son **UNDERLINE**, que indica que el texto se muestre subrayado por debajo del texto, **OVERLINE**, que indica que el texto se muestre subrayado por encima del texto y **LINE-THROUGH**, que indica que el texto se muestre tachado.

4.6.1.10 PROPIEDAD TEXT-INDENT

Especifica la sangría que se debe aplicar a la primera línea del texto. Entre sus posibles valores podemos encontrar **[PORCENTAJE]**, que indica un tanto por ciento de sangría con respecto al ancho del elemento actual y un **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS anteriormente comentadas.

NOTA

Aunque no es frecuente, el valor de la sangría puede ser negativo. Si esto es así, el texto se sangrará a la izquierda en vez de hacia la derecha.

4.6.1.11 PROPIEDAD TEXT-TRANSFORM

Especifica la capitalización del texto. Entre sus posibles valores, los más recurrentes y frecuentemente utilizados tenemos **NONE**, que indica que no se debe capitalizar el texto y es el valor por defecto, **CAPITALIZE**, que indica que se capitalice el primer carácter de cada palabra, **UPPERCASE**, que indica que se transforme todo el texto a mayúsculas y **LOWERCASE**, que indica que se transforme todo el texto a minúsculas.

4.6.1.12 PROPIEDAD VERTICAL-ALIGN

Especifica la alineación vertical para el elemento y es particularmente útil cuando se trabaja con texto e imágenes. Entre sus posibles valores podemos encontrar múltiples configuraciones, sin embargo, las más frecuentes son **TOP**, que indica que el texto debe alinearse en la parte superior con respecto al elemento más alto de la línea y **MIDDLE**, que indica que el texto debe alinearse en el medio del elemento padre.

4.6.1.13 PROPIEDAD WHITE-SPACE

Especifica como se debe gestionar el espacio en blanco en el elemento textual actual. Entre sus posibles valores podemos encontrar múltiples configuraciones, sin embargo,

las más frecuentes son **NORMAL**, que es el valor por defecto e indica que las secuencias de espacios en blanco y tabulaciones se reducirán a un único elemento, haciendo que el texto salte a la línea siguiente cuando sea necesario y, **NOWRAP**, que indica que las secuencias de espacios en blanco y tabulaciones se reducirán a un único elemento, pero el texto nunca saltará a la siguiente línea hasta que encuentre un elemento BR.

4.6.1.14 PROPIEDAD WORD-BREAK

Especifica como se deben cortar las palabras cuando no entran en el espacio asignado al elemento. Entre sus posibles valores podemos encontrar múltiples configuraciones, sin embargo, las más frecuentes son **NORMAL**, que es el valor por defecto e indica que no se produzcan cortes en las palabras, provocando que la palabra entera caiga hacia la siguiente línea y **BREAK-ALL**, que indica que las palabras deben cortarse por cualquier carácter para que entren en el espacio del elemento.

4.6.1.15 PROPIEDAD WORD-SPACING

Especifica el espacio que debe haber entre las palabras del texto. Entre sus posibles valores podemos encontrar **NORMAL**, que indica que las palabras deben estar separadas por un espacio equivalente a 0.25EM y es el valor por defecto y, **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.1.16 PROPIEDAD WORD-WRAP

Especifica que las palabras del texto pueden ajustarse o separarse a la siguiente línea. Entre sus posibles valores podemos encontrar **NORMAL**, que indica que se deben cortar las palabras sólo en los puntos de ruptura permitidos, lo que se suele convertir en que no se corten y es el valor por defecto y, **BREAK-WORD**, que indica que se pueden cortar las palabras en cualquier punto.

4.6.2 Listas

4.6.2.1 PROPIEDAD LIST-STYLE

Es una propiedad compuesta que especifica el tipo de lista que se desea utilizar en el elemento. Se compone de un valor de estilo de un parámetro que se corresponde con la propiedad **LIST-STYLE-IMAGE**, la cual permite definir la imagen que desea utilizar como viñetas. Una posición, que se corresponde con la propiedad **LIST-STYLE-POSITION**, la cual permite definir la posición de la viñeta y, una posible imagen, que se corresponde con la propiedad **LIST-STYLE-TYPE**, la cual permite definir el tipo de viñeta cuando no se utiliza una imagen como marcador. No obstante, este último no suele usarse nunca.

Dado que se van a comentar todas y cada una de las propiedades independientes que pueden asignarse a esta propiedad compuesta, sólo expondremos un ejemplo.

```
ul { list-style: circle inside url("circle.png"); }
```

4.6.3 Márgenes internos y externos

Aunque se haga referencia con terminología similar en nuestro idioma, el uso de los márgenes internos y externos tiene diferentes objetivos. Mientras que el margen interno puede estar enfocado a provocar un efecto decorativo, el externo puede estar enfocado a provocar una mejor legibilidad.

4.6.3.1 PROPIEDADES MARGIN Y PADDING

Las propiedades **MARGIN** y **PADDING** son unas propiedades compuestas que especifican el margen externo e interno respectivamente.

Ambas, admiten entre uno y cuatro valores que el agente de usuario interpretará de manera distinta, dependiendo del número de estos. Entre sus posibles valores podemos encontrar **AUTO**, que indica que se debe dejar que los valores sean provistos por el agente de usuario y, únicamente, es aplicable a **MARGIN** y, **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS.

Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisiéramos asignar 15PX de margen externo a los elementos LI en todos sus lados podríamos hacer:

```
li { margin: 15px 15px 15px 15px; }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor a los lados y valor inferior. Por ejemplo, si quisiéramos asignar 1EM de margen interno a las partes superior e inferior y 0.5EM a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { padding: 1em 0.5em 1em; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos eliminar el margen externo superior e inferior de los elementos DIV y asignar valores automáticos a los laterales, podríamos hacer:

```
div { margin: 0 auto; }
```

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor para el margen superior, para el margen derecho, para el margen inferior y para el margen izquierdo. Por ejemplo, si quisiéramos asignar 15PX de margen interno a los elementos LI en todos sus lados podríamos hacer:

```
li { padding: 15px; }
```

En lo referente a los valores por defecto, la propiedad **MARGIN** tiene asignado el valor **AUTO** a todos sus lados y, la propiedad **PADDING** tiene asignado el valor **0** a todos sus lados.

Además de la compuesta, CSS permite especificar los valores de margen de forma independiente a través de la adición de los sufijos **TOP**, **RIGHT**, **BOTTOM** y **LEFT**. La

concatenación de las palabras clave MARGIN o PADDING con uno de estos sufijos, provocará que se asigne el valor para el lado especificado.

```
ul { margin-top: 2ch; }
li { margin-right: auto; }
p { padding-bottom: 1vw; }
div { padding-left: 1rem; }
```

4.6.4 Bordes internos y externos

Aunque se haga referencia con terminología similar en nuestro idioma, el uso de los bordes internos y externos tiene diferentes objetivos. Mientras que el margen interno puede estar enfocado a provocar un efecto decorativo, el externo puede estar enfocado a provocar una mejor legibilidad y localización.

4.6.4.1 PROPIEDADES BORDER Y OUTLINE

Las propiedades BORDER y OUTLINE son unas propiedades compuestas que especifican el margen externo e interno respectivamente.

Ambas, son propiedades compuestas que se definen a través de un valor de ancho de borde, un estilo de borde y un color de borde.

Todas estas características pueden asignarse a través de sus propiedades individuales BORDER-COLOR o OUTLINE-COLOR, que permiten asignar el color del borde interno o externo respectivamente, BORDER-STYLE o OUTLINE-STYLE, que permiten asignar el estilo del borde interno o externo y BORDER-WIDTH o OUTLINE-COLOR, que permiten asignar el ancho o tamaño de cada uno de los bordes internos o externos.

La secuencia de asignación para esta propiedad es ancho, estilo y color, aunque es posible que los navegadores entiendan otro posible orden de asignación.

```
div { border: 2px solid #f0f0f0; }
div { outline: 2px solid #f0f0f0; }
```

4.6.4.2 PROPIEDADES BORDER-COLOR Y OUTLINE-COLOR

La propiedad BORDER-COLOR es una propiedad compuesta que especifica el color de los bordes del elemento. Al igual que sucede con los márgenes internos y externos, el orden de asignación es valor superior, valor derecho, valor inferior y valor izquierdo. Entre sus posibles valores podemos encontrar TRANSPARENT, que indica que los bordes deben ser de color transparente y [COLOR], que es una palabra clave como pueda ser uno de los valores preestablecidos por los navegadores o, un código de color en hexadecimal, RGBA o HSLA.

Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisieramos asignar un color rojo a los elementos LI en todos sus lados podríamos hacer:

```
li { border-color: red #F00 #FF0000 rgba(255, 0, 0); }
li { outline-color: red #F00 #FF0000 rgba(255, 0, 0); }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor a los lados y valor inferior. Por ejemplo, si quisiéramos asignar un color azul claro a los lados superior e inferior y un color azul oscuro a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { border-color: lightblue darkblue lightblue; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos asignar un color negro a los lados superior e inferior y un color gris a los lados derecho e izquierdo en todos los elementos DIV podríamos hacer:

```
div { border-color: black gray; }
```

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor al borde superior, borde derecho, borde inferior y borde izquierdo. Por ejemplo, si quisiéramos asignar un color naranja a los elementos LI en todos sus lados podríamos hacer:

```
li { border-color: orange; }
```

Por último, cabe mencionar que, además de la compuesta, CSS permite especificar los valores de forma independiente a través de la adición de los prefijos TOP, RIGHT, BOTTOM y LEFT. La construcción de BORDER-COLOR con estos nombres, provocará que se asigne el valor para el lado especificado.

```
div { border-top-color: rgb(0, 0, 0); }
div { border-right-color: rgba(0, 0, 0, 1); }
div { border-bottom-color: hsl(0, 0%, 0%); }
div { border-left-color: hsla(0, 0%, 0%, 1); }
```

En lo referente a la propiedad OUTLINE-COLOR es idéntica a BORDER-COLOR, con la salvedad de que, OUTLINE-COLOR, sólo admite un valor de color para todos los bordes, en vez de cuatro.

4.6.4.3 PROPIEDAD BORDER-STYLE Y OUTLINE-STYLE

Es una propiedad compuesta que especifica el estilo de los bordes del elemento. Al igual que sucede con otras propiedades, el orden de asignación es valor superior, valor derecho, valor inferior y valor izquierdo. Entre sus posibles valores podemos encontrar:

Valor	Descripción	Muestra
none	Indica que NO se desean bordes en ninguno de los lados.	
hidden	Indica que los bordes deben estar ocultos.	
dotted	Indica que los bordes deben ser únicos y punteados.	
dashed	Indica que los bordes deben ser únicos y rayados.	
solid	Indica que los bordes deben ser únicos y continuos.	
double	Indica que los bordes deben ser dobles y continuos.	
groove	Indica que los bordes deben tener un efecto 3D acanalado.	
ridge	Indica que los bordes deben tener un efecto 3D ondulado.	
inset	Indica que los bordes deben tener un efecto 3D presionado.	
outset	Indica que los bordes deben tener un efecto 3D en relieve.	

Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisiéramos asignar un estilo sólido a los elementos LI en todos sus lados podríamos hacer:

```
li { border-style: solid solid solid solid ; }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor derecho e izquierdo y valor inferior. Por ejemplo, si quisiéramos asignar un estilo de borde sólido a los lados superior e inferior y uno punteado a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { border-style: solid dotted solid; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos asignar un estilo rayado a los lados superior e inferior y uno punteado a los lados derecho e izquierdo en todos los elementos DIV podríamos hacer:

```
div { border-style: dashed dotted; }
```

Si se asigna un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor al borde superior, borde derecho, borde inferior y borde izquierdo. Por ejemplo, si quisiéramos asignar un estilo 3D ondulado a los elementos LI en todos sus lados podríamos hacer:

```
li { border-style: ridge; }
```

Por último, cabe mencionar que, además de la compuesta, CSS permite especificar los valores de forma independiente a través de la adición de los prefijos TOP, RIGHT, BOTTOM y LEFT. La construcción de BORDER-STYLE con estos nombres, provocará que se asigne el valor para el lado especificado.

```
div { border-top-style: solid; }
div { border-right-style: solid; }
div { border-bottom-style: solid; }
div { border-left-style: solid; }
```

En lo referente a la propiedad OUTLINE-STYLE es idéntica a BORDER-STYLE, con la salvedad de que, OUTLINE-STYLE, sólo admite un valor de estilo para todos los bordes, en vez de cuatro.

4.6.4.4 PROPIEDADES BORDER-WIDTH Y OUTLINE-WIDTH

La propiedad BORDER-WIDTH es una propiedad compuesta que especifica el tamaño de los bordes del elemento. Al igual que sucede con los márgenes internos y externos, y otras propiedades de su mismo contexto, el orden de asignación es valor superior, valor derecho, valor inferior y valor izquierdo. Entre sus posibles valores podemos encontrar THIN, que indica que los bordes deben tener un tamaño fino o delgado, equivalente a 1 píxel, MEDIUM, que es el valor por defecto e indica que los bordes deben tener un tamaño medio, equivalente a 3 píxeles, THICK, que indica que los bordes deben tener un tamaño grueso, equivalente a 5 píxeles y [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

Si se declaran los cuatro valores, la secuencia de asignación será valor superior, valor derecho, valor inferior y valor izquierdo. Por ejemplo, si quisiéramos asignar un tamaño de borde de 2PX a los elementos LI en todos sus lados podríamos hacer:

```
li { border-width: 2px 2px 2px 2px; }
```

Si se declaran tres valores, la secuencia de asignación será valor superior, valor derecho e izquierdo y valor inferior. Por ejemplo, si quisiéramos asignar un tamaño de borde de 2PX a los lados superior e inferior y un tamaño de borde de 1PX a los lados derecho e izquierdo en todos los elementos P podríamos hacer:

```
p { border-width: 2px 1px 2px; }
```

Si se declaran dos valores, la secuencia de asignación será valor superior e inferior y valor derecho e izquierdo. Por ejemplo, si quisiéramos asignar un tamaño de borde

de 2PX a los lados superior e inferior y un tamaño de borde de 4PX a los lados derecho e izquierdo en todos los elementos DIV podríamos hacer:

```
div { border-width: 2px 4px; }
```

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor al borde superior, borde derecho, borde inferior y borde izquierdo. Por ejemplo, si quisiéramos asignar un tamaño de borde de 2PX a los elementos LI en todos sus lados podríamos hacer:

```
li { border-width: 2px; }
```

Por último, cabe mencionar que, además de la compuesta, CSS permite especificar los valores de forma independiente a través de la adición de los prefijos TOP, RIGHT, BOTTOM y LEFT. La construcción de BORDER-WIDTH con estos nombres, provocará que se asigne el valor para el lado especificado.

```
div { border-top-width: 2px; }
div { border-right-width: 1px; }
div { border-bottom-width: 1px; }
div { border-left-width: 2px; }
```

En lo referente a la propiedad OUTLINE-WIDTH es idéntica a BORDER-WIDTH, con la salvedad de que, OUTLINE-WIDTH, sólo admite un valor de estilo para todos los bordes, en vez de cuatro.

4.6.4.5 PROPIEDAD BORDER-RADIUS

Es una propiedad compuesta que especifica el tamaño de la curva (o radio) que une los bordes del elemento. Para esta propiedad, el orden de asignación es valor esquina superior izquierda, valor esquina superior derecha, valor esquina inferior derecha y valor esquina inferior izquierda.

Sus posibles valores sólo pueden asignarse a través de un valor establecido en una de las medidas permitidas de CSS.

Si se declaran los cuatro valores, la secuencia de asignación será la anteriormente comentada. Así, por ejemplo, si quisiéramos asignar un radio de borde de 15PX a los elementos LI en todas sus esquinas podríamos hacer:

```
li { border-radius: 15px 15px 15px 15px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15PX 15PX 15PX 15PX

Si se declaran tres valores, la secuencia de asignación será valor esquina superior izquierda, valor esquinas superior derecha e inferior izquierda y valor esquina inferior derecha. Por ejemplo, si quisiéramos asignar un radio de borde de 15PX a las esquinas

superior izquierda e inferior derecha y un radio de borde de 0PX a las esquinas superior derecha e inferior izquierda en todos los elementos P podríamos hacer:

```
p { border-radius: 15px 0px 15px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15PX 0PX 15PX

Si se declaran dos valores, la secuencia de asignación será valor esquinas superior izquierda e inferior derecha y valor esquinas superior derecha e inferior izquierda. Por ejemplo, si quisiéramos asignar un radio de borde de 15PX a las esquinas superior izquierda e inferior derecha y un radio de borde de 0PX a las esquinas superior derecha e inferior izquierda en todos los elementos DIV podríamos hacer:

```
div { border-width: 15px 0px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15PX 0

Si se declara un único valor, la secuencia de asignación será para todos los valores, es decir, se asignará el mismo valor para las esquinas superior izquierda, superior derecha, inferior derecha e inferior izquierda. Por ejemplo, si quisiéramos asignar un radio de borde de 15PX a los elementos LI en todas sus esquinas podríamos hacer:

```
li { border-width: 15px; }
```

Y si lo ejecutásemos, el resultado sería algo como:

ESTE ELEMENTO TIENE UN BORDER-RADIUS DE 15 PÍXELES

4.6.4.6 PROPIEDAD OUTLINE-OFFSET

Especifica el espacio en blanco que debe existir entre el borde interno proporcionado por la propiedad BORDER y el borde externo proporcionado por la propiedad OUTLINE. Entre sus posibles valores podemos encontrar [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.5 Colores

4.6.5.1 CODIFICACIÓN DE COLORES

Los colores en HTML pueden establecerse a través de los nombres preestablecidos por el agente de usuario, mediante codificación RGB, HSL o hexadecimal. Tanto RGB, como HSL, admiten una variación que permite especificar el canal alfa.

4.6.5.1.1 Codificación RGB y RGBA

El modelo de color RGB (Red - Green - Blue) es un modelo matemático abstracto que describe cómo realizar la codificación o representación de colores a partir de unos componentes de intensidad sobre los colores primarios. Dado que su formulación es únicamente a través de rojo, verde, azul, la codificación RGB no presenta la posibilidad de definir un color con transparencia.

El espacio de color RGBA (Red - Green - Blue - Alpha) es una combinación del modelo RGB con un cuarto componente denominado canal de alfa. Este canal alfa es el que posibilita que los colores puedan tener una transparencia.

El valor de los componentes rojo, verde y azul se establecen a partir de un valor binario de 8 bits bajo el sistema decimal, es decir, con valores que van de 0 a 255. Sin embargo, el valor del canal alfa se establece con un valor en tanto por uno.

```
RGB(0, 0, 255); /* Color azul al 50% de transparencia */  
RGBA(0, 0, 255, 0.5); /* Color azul al 50% de transparencia */
```

4.6.5.1.2 Codificación Hexadecimal

La codificación de colores en hexadecimal funciona exactamente igual que el modelo RGB, es decir, los colores pueden formalizarse o construirse a partir de unos valores de rojo, verde, azul, no obstante, su anotación es diferente.

Cabe destacar que existen algunos navegadores que permiten definir colores en hexadecimal y con transparencia a través de un código de ocho caracteres hexadecimales, en vez de seis. No obstante, este tipo de codificación con transparencia puede no ser compatible con algunos navegadores.

El valor de los componentes rojo, verde y azul y el canal alfa van de 00 a FF y, aunque en general, esta especificación se aplica mediante la concatenación del carácter almohadilla y una cadena de seis caracteres, también es posible codificar los colores mediante una cadena de longitud tres.

Cuando se utiliza este tipo de anotación, el color generado se construye a partir de la duplicación y anidación de cada carácter, es decir, cuando se especifica un color codificado como 0F0, en realidad, lo que se está generando es el color 0OFF00, en este caso un verde.

4.6.5.1.3 Codificación HSL y HSLA

El modelo de color HSL (Hue - Saturation - Lightness) es un modelo matemático abstracto que describe cómo realizar la codificación o representación de colores a partir de unos componentes de matiz, saturación y luminosidad. Dado que su formulación es únicamente a través de estos tres componentes, la codificación HSL no presenta la posibilidad de definir un color con transparencia.

El espacio de color HSLA (Hue - Saturation - Lightness - Alpha) es una combinación del modelo HSL con un cuarto componente denominado canal de alfa. Este canal alfa es el que posibilita que los colores puedan tener una transparencia.

El componente de matiz se mide en grados, por lo que sus posibles valores van de 0 a 360. El valor 0 equivale al rojo, 120 al verde y 240 al azul.

Los componentes de saturación y luminosidad se establecen en términos porcentaje, es decir, con valores que van de 0 a 100 y, el canal alfa, se establece en términos de tanto por uno. Un ejemplo de esta codificación es `HSL(0, 240%, 50%)`, el cual equivale a decir que es un azul al 50% de transparencia.

4.6.5.2 PROPIEDADES PARA EL COLOR

4.6.5.2.1 Propiedad background

Aunque la propiedad BACKGROUND admite múltiples opciones, uno de los usos más extendidos es, únicamente, la asignación de un color de fondo o degradado. De hecho, su definición permite establecer un color de fondo, una imagen o efecto gradiente, una posición, un tamaño, una opción de repetición, un origen, una opción de extensión y una de desplazamiento.

Todos estos posibles valores son interpretados de forma unívoca e independiente y, por decirlo así, todos son opcionales, lo cual permite que se puedan asignar uno o varios valores en cualquier orden.

Al igual que sucede con otras propiedades, también permite la asignación a través de sus propiedades individuales y, aunque todas ellas se verán, de forma independiente, en el capítulo de imágenes y multimedia, aquí las comentaremos por encima, sobre todo, para obtener una idea de su capacidad y utilidad.

Valor	Descripción
<code>background-attachment</code>	Define opciones de desplazamiento.
<code>background-color</code>	Define el color del fondo.
<code>background-clip</code>	Define las opciones de extensión para el fondo.
<code>background-image</code>	Define la imagen o efecto gradiente.
<code>background-origin</code>	Define el origen de la imagen o fondo.
<code>background-position</code>	Define la posición de la imagen o fondo.
<code>background-repeat</code>	Define las opciones de repetición del fondo.
<code>background-size</code>	Define el tamaño de la imagen o fondo.

Como se puede observar, su sintaxis es bastante compleja, sin embargo, su uso está muy recomendado, debido, fundamentalmente, a que está diseñada para ahorrar tiempo de proceso y carga.

```
div { background: purple; }
div { background: url("./imagen-fondo.png) repeat-x; }
div { background: padding-box black; }
div { background: no-repeat center/cover url("./imagen-fondo.png); }
li { background: #FFF url("plus.png") no-repeat fixed left center; }
```

4.6.5.2.2 Propiedad background-color

Especifica el color del fondo que será aplicado en el elemento, el cual afecta (o incluye) el espacio asignado al margen interno y al borde, pero no al margen externo. Entre sus posibles valores podemos encontrar [COLOR], que indica el color de fondo en formato HSL, RGBA o hexadecimal y TRANSPARENT, que indica que el color de fondo es transparente y es equivalente a RGBA (0, 0, 0, 0).

```
div { background-color: rgba(128, 0, 128, 1); }
```

4.6.5.2.3 Propiedad color

Especifica el color del texto. Esta propiedad admite una larga lista de colores a través de su nombre en inglés o mediante su equivalente en RGB(A), HSL(A) o hexadecimal.

4.6.5.2.4 Propiedades text-shadow y box-shadow

Si la propiedad TEXT-SHADOW es una propiedad compuesta que agrega una sombra al texto, la propiedad BOX-SHADOW es una propiedad compuesta que agrega una sombra a los elementos no textuales. Su sintaxis es:

```
text-shadow: PosX PosY radio_desenfoque color;
box-shadow: inset PosX PosY radio_desenfoque radio_propagación color;
```

Si el valor de la PosX es positivo, la sombra avanzará en sentido hacia la derecha, por lo que, si es negativo, avanzará en sentido hacia la izquierda. Algo similar pasa con el segundo parámetro. Si el valor de PosY es positivo, la sombra avanzará en sentido hacia abajo, por lo que, si es negativo, avanzará en sentido hacia arriba.

Para los valores de desenfoque, cuanto mayor sean sus valores, mayor será la difuminación y expansión de la sombra. En lo referente a los valores del parámetro de color, admite una larga lista de opciones a través de su nombre en inglés o mediante su equivalente en RGB(A), HSL(A) o hexadecimal.

NOTA

Salvo excepciones, las propiedades TEXT-SHADOW y BOX-SHADOW no se deben utilizar porque, además de dificultar su lectura y disminuir la legibilidad, puede disminuir la accesibilidad y proporcionar una imagen corporativa "desaliñada".

4.6.6 Posicionamiento

4.6.6.1 PROPIEDAD CLEAR

Especifica en qué lado debe romperse la línea de posicionamiento flotante proporcionada por la propiedad FLOAT. El efecto de romper el posicionamiento flotante viene a significar que, los elementos que estén por la parte que indica esta propiedad, caerán hacia la línea siguiente. Entre sus posibles valores podemos encontrar **BOTH**, que indica que debe romperse por ambos lados (independientemente del valor de la propiedad FLOAT), **LEFT**, que indica que romperse por el lado izquierdo (siempre y cuando el valor de la propiedad FLOAT sea LEFT), **NONE**, que es el valor por defecto e indica que se permiten elementos flotantes a ambos lados del elemento actual y **RIGHT**, que indica que romperse por el lado derecho (siempre y cuando el valor de la propiedad FLOAT sea RIGHT).

Por dejar el tema algo más claro, por ejemplo, si tenemos tres elementos con la propiedad FLOAT establecida a LEFT o RIGHT, y establecemos la propiedad CLEAR a BOTH en todos ellos, cada elemento debería posicionarse en una nueva línea.

Ahora, si esos mismos elementos tienen la propiedad FLOAT establecida a LEFT o RIGHT, y establecemos la propiedad CLEAR al mismo valor que FLOAT en el segundo, el primero debería permanecer en la línea de posicionamiento actual y, el resto, deberían posicionarse en una nueva línea. Sin embargo, si al segundo elemento se le establece la propiedad CLEAR a otro valor que no sea el de FLOAT, no tendrá ningún efecto.

```
aside { clear: right; }
div { clear: both; }
p { clear: left; }
```

NOTA

Los elementos que tengan un posicionamiento absoluto, es decir, tengan la propiedad POSITION establecida a ABSOLUTE, ignorarán esta propiedad.

4.6.6.2 PROPIEDAD FLOAT

Especifica un posicionamiento flotante para el elemento. Entre sus posibles valores podemos encontrar **LEFT**, que indica que el elemento debe estar flotando a la izquierda, **NONE**, que es el valor por defecto e indica que NO debe flotar, es decir, que se mostrará justo dónde aparezca el elemento y **RIGHT**, que indica que el elemento debe estar flotando a la derecha.

```
aside { float: right; }
div { float: left; }
p { float: none; }
```

NOTA

Los elementos que tengan un posicionamiento absoluto, es decir, tengan la propiedad POSITION establecida a ABSOLUTE, ignorarán esta propiedad.

4.6.6.3 PROPIEDAD POSITION

Especifica el tipo de posicionamiento utilizado para el elemento. Entre sus posibles valores podemos encontrar:

Valor	Descripción
absolute	Indica que el elemento se posiciona con respecto al primer ancestro que tenga un posicionamiento relativo o estático.
fixed	Indica que el elemento se posiciona con respecto a la ventana del navegador.
relative	Indica que el elemento debe posicionarse de forma relativa con respecto a su elemento hermano o anterior.
static	Indica que los elementos se procesan y posicionan en el orden en el que llegan o aparecen. Es el valor por defecto.
sticky	Sólo es efectivo cuando el elemento presenta una barra de desplazamiento e indica que el elemento debe tener un comportamiento mixto de relativo y fijo. Por ejemplo, si un elemento contenedor tiene la barra de desplazamiento habilitada y uno de sus elementos hijo se establece como STICKY, mientras esté por encima del punto indicado por la propiedad TOP, se comportará como si tuviese un posicionamiento relativo. Sin embargo, en el momento en que se sobrepase el valor establecido por TOP, se quedará adherido como si tuviese posicionamiento fijo.

Como norma, cuando se trabaja con posicionamientos absolutos, la coordenada (0,0) del elemento dependerá de su último ancestro con posicionamiento estático o relativo. Sin embargo, cuando se trabaja con posicionamientos fijos, la coordenada (0,0) siempre se corresponderá con la coordenada (0,0) del documento.

4.6.6.4 PROPIEDAD BOTTOM

Especifica que el elemento debe estar posicionado con respecto a su propiedad inferior. Entre sus posibles valores podemos encontrar **AUTO**, que indica que el navegador es quien debe calcular la posición y es el valor por defecto y **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad BOTTOM es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es ABSOLUTE, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad BOTTOM de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es ABSOLUTE, FIXED o RELATIVE, su desplazamiento será hacia arriba o hacia abajo en función de si es o no positivo, respectivamente.

Si el posicionamiento es STICKY (adherido), se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo. Ahora bien, si su posicionamiento es STATIC, la propiedad BOTTOM no tendrá ningún efecto.

4.6.6.5 PROPIEDAD LEFT

Especifica que el elemento debe estar posicionado con respecto a su propiedad izquierda. Entre sus posibles valores podemos encontrar AUTO, que indica que el navegador es quién debe calcular la posición y es el valor por defecto y, [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad LEFT es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es ABSOLUTE, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad LEFT de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es ABSOLUTE, FIXED o RELATIVE, su desplazamiento será hacia la derecha o hacia la izquierda en función de si es o no positivo, respectivamente.

Si el posicionamiento es STICKY (adherido) se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo. Eso sí, sólo funcionará si la barra de desplazamiento horizontal está visible y habilitada. Ahora bien, si su posicionamiento es STATIC, la propiedad LEFT no tendrá ningún efecto.

4.6.6.6 PROPIEDAD RIGHT

Especifica que el elemento debe estar posicionado con respecto a su propiedad derecha. Entre sus posibles valores podemos encontrar AUTO, que indica que el navegador es quién debe calcular la posición y es el valor por defecto y, [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad RIGHT es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es ABSOLUTE, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad RIGHT de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es ABSOLUTE, FIXED o RELATIVE, su desplazamiento será hacia la izquierda o hacia la derecha en función de si es o no positivo, respectivamente.

Si el posicionamiento es STICKY (adherido) se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo. Eso sí, sólo funcionará si la barra de desplazamiento horizontal está visible y habilitada. Ahora bien, si su posicionamiento es STATIC, la propiedad RIGHT no tendrá ningún efecto.

4.6.6.7 PROPIEDAD TOP

Especifica que el elemento debe estar posicionado con respecto a su propiedad superior. Entre sus posibles valores podemos encontrar **AUTO**, que indica que el navegador es quién debe calcular la posición y es el valor por defecto y, **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS.

La propiedad **TOP** es muy sencilla, no obstante, posee unas reglas que hay que tener claras. Por ejemplo, si el posicionamiento es **ABSOLUTE**, el punto de origen desde el cual, el elemento se empezará a desplazar, será el equivalente al valor de la propiedad **TOP** de su primer ancestro con posicionamiento relativo.

Si el posicionamiento es **ABSOLUTE**, **FIXED** o **RELATIVE**, su desplazamiento será hacia abajo o hacia arriba en función de si es o no positivo, respectivamente. Eso sí, esto sólo funcionará si la barra de desplazamiento vertical está visible y habilitada.

Si el posicionamiento es **STICKY** (adherido), se comportará como si tuviese un posicionamiento relativo hasta que llegue a ese valor establecido. A partir de ese momento, se quedará adherido como si tuviese posicionamiento fijo. Ahora bien, si su posicionamiento es **STATIC**, la propiedad **TOP** no tendrá ningún efecto.

4.6.6.8 PROPIEDAD Z-INDEX

Especifica el orden de apilamiento del elemento. Entre sus posibles valores podemos encontrar **AUTO**, que indica que el navegador es quién debe calcular la posición y es el valor por defecto y, **[NÚMERO]**, que es un valor entero que indica el orden de presentación.

Si el valor es positivo, se presentará según el orden de apilamiento. Esto es, cuanto mayor sea el valor, más arriba de la pila estará y más posibilidades de mostrarse primero tendrá.

Sin embargo, si el valor es negativo, se establecerá por detrás del orden de apilamiento general de sus hermanos o ancestros, por lo que puede que se oculte o vuelva invisible.

NOTA

La propiedad **Z-INDEX** solamente tendrá efecto cuando el posicionamiento sea absoluto, relativo o fijo.

4.6.7 Comportamientos y tamaños

4.6.7.1 PROPIEDAD BOX-SIZING

Especifica cómo deben asignarse y calcularse el alto y ancho de los elementos. Esto es, si deben incluir los márgenes internos (padding) y/o los bordes, o no. Entre sus posibles valores podemos encontrar **CONTENT-BOX**, que indica que se debe incluir sólo el contenido, e ignorar los márgenes internos y bordes y es el valor por defecto y, **BORDER-BOX**, que indica que se deben incluir el contenido, padding y bordes.

NOTA

En general, se puede afirmar que, trabajar con cajas o capas incluyendo los márgenes internos y los bordes es más fácil de manejar y facilita el diseño adaptativo, aunque no siempre.

4.6.7.2 PROPIEDAD DISPLAY

Especifica cómo se debe representar el elemento. Entre sus posibles valores podemos encontrar:

Valor	Descripción
block	Indica que el elemento debe mostrarse en bloque. Esto es, el elemento comenzará en una nueva línea y que ocupará todo el ancho disponible.
contents	Indica que el elemento debe mostrarse como si fuese un contenido. Esto significa que se pierde el concepto de caja o capa y que, en su lugar, será reemplazado por un contenedor virtual que contiene sus elementos descendientes.
flex	Indica que el elemento debe comportarse como un elemento de bloque flexible. Esto es, los elementos se recolocarán en cualquier dirección y podrán ampliar o reducir sus tamaños en función del espacio disponible para llenarlo de forma eficiente sin provocar desbordamientos ni cortes.
grid	Indica que el elemento debe comportarse como un elemento de bloque de una cuadrícula, similar a una tabla, por lo que los elementos se recolocarán a modo de filas y columnas.
inline	Indica que el elemento debe mostrarse en línea. Esto conllevará que las propiedades de WIDTH y HEIGHT sean ignoradas y, por lo tanto, no tengan ningún efecto.
inline-block	Indica que el elemento debe mostrarse como un conjunto de bloques en línea. Esto es, el elemento se trata como si estuviese en INLINE , pero admite la aplicación de las propiedades de WIDTH y HEIGHT .
inline-flex	Indica que el elemento debe comportarse como un contenedor flexible, aunque con el concepto de línea, en vez de en bloque.
inline-grid	Indica que el elemento debe comportarse como un conjunto de contenedores de cuadrícula en línea, en vez de en bloque.
inline-table	Indica que el elemento debe comportarse como si fuese un conjunto de tablas en línea, en vez de en bloque.
list-item	Indica que el elemento debe comportarse como si fuese una lista, con sus viñetas y demás características.
none	Indica que el elemento NO debe mostrarse.

table	Indica que el elemento debe comportarse como una matriz bidimensional compuesta por filas y columnas.
table-caption	Indica que el elemento debe comportarse como si fuese el elemento CAPTION de una tabla.
table-cell	Indica que el elemento debe comportarse como si fuese la celda de una tabla, es decir, como el elemento TD.
table-column	Indica que el elemento debe comportarse como si fuese la columna de una tabla, es decir, como el elemento COL.
table-column-group	Indica que el elemento debe comportarse como si de un grupo de columnas de una tabla se tratase, es decir, como el elemento COLGROUP.
table-header-group	Indica que el elemento debe comportarse como si fuese la cabecera de una tabla, es decir, como el elemento THEAD.
table-footer-group	Indica que el elemento debe comportarse como si fuese el pie de una tabla, es decir, como el elemento TFOOTER.
table-row	Indica que el elemento debe comportarse como si fuese la fila de una tabla, es decir, como el elemento TR.
table-row-group	Indica que el elemento debe comportarse como si fuese el cuerpo de una tabla, es decir, como el elemento TBODY.

NOTA

Mientras que, en documentos HTML, el valor por defecto suele ser BLOCK, en los documentos XML y SVG suele ser INLINE.

4.6.7.3 PROPIEDAD HEIGHT

Especifica la altura del elemento. Entre sus posibles valores podemos encontrar AUTO, que indica que el navegador es quién debe calcular y asignar el alto y es el valor por defecto y, [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.7.4 PROPIEDAD MAX-HEIGHT

Especifica la altura máxima del elemento. Entre sus posibles valores podemos encontrar NONE, que indica que no hay máximo establecido y es el valor por defecto y, [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.7.5 PROPIEDAD MAX-WIDTH

Especifica la anchura máxima del elemento. Entre sus posibles valores podemos encontrar NONE, que indica que no hay máximo establecido y es el valor por defecto y, [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.7.6 PROPIEDAD MIN-HEIGHT

Especifica la altura mínima del elemento. Entre sus posibles valores podemos encontrar **AUTO**, que indica que no hay mínimo establecido y es el valor por defecto y, **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.7.7 PROPIEDAD MIN-WIDTH

Especifica la anchura mínima del elemento. Entre sus posibles valores podemos encontrar **AUTO**, que indica que no hay mínimo establecido y es el valor por defecto y, **[VALOR]**, que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.7.8 PROPIEDAD OPACITY

Especifica la opacidad del elemento. Entre sus posibles valores podemos encontrar un **[NÚMERO]** y que es un valor decimal que indica, en tanto por uno, el porcentaje de visibilidad del elemento.



Negro al 100%
opacity: 1;



Negro al 50%
opacity: 0.5;



Negro al 5%
opacity: .05;

4.6.7.9 PROPIEDAD OVERFLOW

Especifica cómo se debe comportar el elemento si su contenido se desborda, es decir, si su contenido no entra en el espacio asignado. Entre sus posibles valores podemos encontrar:

- ▶ **AUTO**: indica que las barras de desplazamiento se mostrarán u ocultarán en función de si el contenido desborda o no, y que el contenido nunca debe verse fuera de los límites del elemento.
- ▶ **HIDDEN**: indica que las barras de desplazamiento se mantengan ocultas, independientemente de que el contenido desborde o no, y que el contenido nunca debe verse fuera de los límites del elemento.
- ▶ **SCROLL**: indica que las barras de desplazamiento se mantengan visibles, independientemente de que el contenido desborde o no, y que el contenido nunca debe verse fuera de los límites del elemento.
- ▶ **VISIBLE**: indica que las barras de desplazamiento se mantengan ocultas y que el contenido se muestre, aunque se desborde. Es el valor por defecto.

NOTA

La propiedad OVERFLOW no tiene ningún efecto en elementos que no tengan definida una altura específica.

Por último, cabe mencionar que, existen unas variaciones de esta propiedad que permiten establecer los valores de forma independiente. Estas propiedades son OVERFLOW-X, que establece el comportamiento para los límites derecho e izquierdo del elemento y, OVERFLOW-Y, que establece el comportamiento para los límites superior e inferior del elemento.

4.6.7.10 PROPIEDAD VISIBILITY

Especifica si el elemento debe estar visible o no. Entre sus posibles valores podemos encontrar COLLAPSE, que indica que el elemento debe aparecer oculto y sin ocupar espacio en pantalla. Este valor sólo es aplicable para los elementos TR, TBODY, COL y COLGROUP, HIDDEN, que indica que el elemento debe aparecer oculto, pero sin colapsar el espacio que ocupa y VISIBLE, que es el valor por defecto e indica que el elemento debe aparecer visible.

4.6.7.11 PROPIEDAD WIDTH

Especifica la anchura del elemento. Entre sus posibles valores podemos encontrar AUTO, que indica que el navegador es quien debe calcular y asignar el ancho y es el valor por defecto y, [VALOR], que indica un valor establecido en una de las medidas permitidas de CSS.

4.6.8 Reglas arroba y media queries

4.6.8.1 REGLA CHARSET

Especifica la codificación de caracteres que se debe utilizar en la hoja de estilos.

Aunque generalmente el único valor que se utiliza es UTF-8 por temas de estandarización y compatibilidad, existen otros muchos valores que puede admitir, disponibles en la web de IANA. Para consultar todos los posibles valores puede visitarse la URL <https://www.iana.org/assignments/character-sets/character-sets.xhtml>. No obstante, para que esta regla funcione, o se aplique, se deben tener en cuenta unas romas:

- La regla arroba CHARSET debe ser la primera declaración de la hoja de estilos.
- Si por cualquier razón, hubiese declaradas o definidas varias reglas CHARSET, la única que tendrá efecto será la primera, el resto serán ignoradas.
- La regla arroba CHARSET no puede estar declarada dentro de ningún atributo o estructura STYLE de HTML.

Ejemplo:

```
@charset "utf-8"
```

4.6.8.2 REGLA FONT-FACE

Especifica una nueva fuente de texto, o nuevo tipo de fuente, para poder ser utilizada como tipografía en el documento o página web.

Cabe destacar que no todos los formatos de fuente están disponibles para todos los navegadores. Así, por ejemplo, los formatos de fuente de texto TTF/OTF y WOFF pueden ser interpretados por la mayoría de los agentes de usuario, pero el formato WOFF2 no está soportado por Internet Explorer ni Safari. En cuanto al formato SVG, es compatible con Safari, pero no con Firefox.

Por esta razón, lo normal es que se utilicen fuentes de texto vectoriales desde algún directorio interactivo de uso público, los cuales cargan todos los posibles formatos para que cada agente de usuario utilice el que más le convenga.

Para que la regla FONT-FACE pueda ser aplicada, se deben definir serie de parámetros u opciones:

Valor	Descripción
font-family	Define el nombre de la fuente que será usado en las reglas CSS para vincular y aplicar el estilo. Es un parámetro requerido.
src	Permite definir la dirección o ubicación del archivo. Es un parámetro requerido.
font-stretch	Permite expandir o condensar el texto. Es opcional y puede contener cualquiera de los posibles valores de FONT-STRETCH.
font-style	Permite establecer el estilo del texto. Es opcional y puede contener cualquiera de los posibles valores de FONT-STYLE.
font-weight	Permite establecer el grosor del texto. Es opcional y puede contener cualquiera de los posibles valores de FONT-WEIGHT.
unicode-range	Permite definir el rango de caracteres Unicode que admite la fuente de texto. El valor por defecto es “U+0-10FFFF”.

Ejemplo:

```
/* Declaración de fuente básica */
@font-face {
  font-family: textFont;
  src: url(new_arial_100.ttf);
  font-weight: 100;
}
```

4.6.8.3 REGLA IMPORT

Permite incluir otras hojas de estilo dentro de la hoja de estilo actual. Debe estar declarada justo después de la declaración de la regla CHARSET.

Ejemplo:

```
@import "custom-styles.css";
```

Aunque, en general, se suelen importar las hojas de estilos sin ningún control de medios o resoluciones, es posible hacerlo en función de estos parámetros.

Ejemplo:

```
@import "mobile.css" screen and (max-width: 768px);
```

Para poder ver todos los posibles valores con los que se puede configurar la regla IMPORT se puede consultar la regla MEDIA, explicada un poco más adelante.

4.6.8.4 REGLA KEYFRAMES

Permite especificar una animación a partir de las reglas CSS que estén contenidas en su declaración. Esta regla, que sólo es aplicable para el contexto de las animaciones CSS, permite cambiar las propiedades y valores durante todo el ciclo de la animación.

Para poder definir el ciclo de vida de la animación se pueden utilizar las palabras clave FROM y TO o recurrir a valores de porcentaje, aunque, siempre es mejor utilizar esta segunda opción por temas de compatibilidad.

NOTA

La característica !IMPORTANT es ignorada dentro de esta regla.

Ejemplo:

```
@keyframes ejemplo1 { from {top: 0px;} to {top: 200px;} }
@keyframes ejemplo2 {
0% {top: 0px; left: 0; }
25% {top: 0px; left: 50px; }
50% {top: 50px; left: 50px}
75% {top: 50px; left: 0}
100% {top: 0; left: 0; }
}
```

Si se desea más información y ejemplos sobre las animaciones se puede ir directamente al capítulo de animaciones, transiciones y efectos.

4.6.8.5 REGLA MEDIA

La regla MEDIA, referida habitualmente como “media query” o consulta de medios, se utiliza para aplicar diferentes reglas de estilo dependiendo de la resolución del dispositivo y/o medio.

Este tipo de consultas puede aplicarse para controlar varias casuísticas como son el medio, ancho y alto de la ventana gráfica (VIEWPORT), ancho y alto del dispositivo, la orientación y/o la resolución.

Seguramente, muchos de los lectores ya se habrán dado cuenta, o ya sabrán, que estas reglas MEDIA son muy utilizadas en diseños adaptativos y/o receptivos tanto en configuraciones de escritorio, como en tables o móviles. No obstante, también son utilizadas, como se ha podido observar en la propiedad IMPORT, para controlar los medios como son la pantalla, la impresora o la voz.

La regla MEDIA funciona de forma similar a como lo hacen otras propiedades de CSS ya que permite una declaración combinada o individual de un tipo de medio, con o sin la especificación de sus características e, incluso, una especificación sin tipo de medio.

Ejemplo:

```
@media screen and (min-width: 768px) and (max-width: 1024px){  
    /* Reglas CSS aplicables para estas circunstancias */  
}
```

Los tipos de medios disponibles son:

- ▶ **ALL**: indica que las reglas contenidas en la consulta de medios son válidas para cualquier tipo de medio. Es el valor por defecto.
- ▶ **PRINT**: indica que las reglas contenidas en la consulta de medios sólo son válidas sólo para el tipo de medio definido como impresora.
- ▶ **SCREEN**: indica que las reglas contenidas en la consulta de medios sólo son válidas para el medio definido como pantalla, independientemente de si pertenece a un dispositivo de escritorio, tablet o móvil.
- ▶ **SPEECH**: indica que las reglas contenidas en la consulta de medios sólo son válidas para agentes de usuario que permiten la lectura mediante control de voz, como los lectores de pantalla utilizados por las personas con discapacidad.

Las características de medios, también llamadas funciones de medios, son muchas y, algunas de ellas, no son compatibles con todos los agentes de usuario, sin embargo, aquí se comentarán la mayoría:

- ▶ **ANY-HOVER**: indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo de entrada disponible puede pasar por los elementos. Sus posibles valores son HOVER, para indicar que todos los dispositivos y mecanismos que tengan disponible esta opción deben aplicar esta

consulta de medios cuando esté disponible el desplazamiento por los elementos o, NONE, para cuando NO esté disponible el desplazamiento por los elementos.

```
@media (any-hover: hover){  
/* Reglas CSS aplicables cuando puede pasar por los elementos */  
}  
  
@media (any-hover: none){  
/* Reglas CSS aplicables cuando NO puede pasar por los elementos */  
}
```

ANY-POINTER: indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo disponga de un dispositivo señalador (o de tipo puntero), sea o no primario, y tenga una precisión determinada. Sus posibles valores son FINE, para indicar que el dispositivo señalador es de alta precisión, COARSE, que indica que el dispositivo señalador es de precisión limitada o, NONE, que indica que no hay dispositivo señalador.

```
@media (any-pointer: coarse){  
/* Reglas CSS aplicables cuando el dispositivo es de baja precisión */  
}  
  
@media (any-pointer: fine){  
/* Reglas CSS aplicables cuando el dispositivo es de alta precisión */  
}  
  
@media (any-pointer: none){  
/* Reglas CSS aplicables cuando no hay dispositivo señalador */  
}
```

ASPECT-RATIO: indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la relación de altura y anchura de la ventana gráfica o VIEWPORT se corresponda con el valor indicado. Además, existen dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (aspect-ratio: 16/9){  
/* Reglas CSS aplicables cuando el dispositivo es 16 a 9 */  
}  
  
@media (min-aspect-ratio: 1/1){  
/* Reglas CSS aplicables cuando el dispositivo es, como máximo, 1 a 1 */  
}  
  
@media (max-aspect-ratio: 11/16){  
/* Reglas CSS aplicables cuando el dispositivo es, como máximo, 11 a 16 */  
}
```

COLOR: indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la profundidad de color (número de bits para representar un

color) solicitada se corresponda con la profundidad de color del dispositivo. Sus posibles valores son 1, 2, 4, 8, 16, 24 y 32 y, por defecto, su valor es 8. Además, existen dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (color){  
/* Reglas CSS aplicables cuando el dispositivo es color */  
}  
  
@media (min-color: 8){  
/* Reglas CSS aplicables cuando el dispositivo admite 8 bits */  
}  
  
@media (max-color: 16){  
/* Reglas CSS aplicables cuando el dispositivo admite 16 bits */  
}
```

COLOR-GAMUT: indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el rango aproximado de colores admitido por el agente de usuario y dispositivo de salida se corresponda con el valor indicado. Sus posibles valores son SRGB, que admite la gama SRGB y son la inmensa mayoría de las pantallas de color, P3, que admite gama especificada por el espacio de color DCI P3 o una superior como SRGB y, REC2020, que admite la gama especificada por la Recomendación UIT-R BT.2020 Color Space o superior.

```
@media (color-gamut: srgb){  
/* Reglas CSS aplicables para estas circunstancias */  
}  
  
@media (color-gamut: p3){  
/* Reglas CSS aplicables para estas circunstancias */  
}  
@media (color-gamut: srgb){  
/* Reglas CSS aplicables para estas circunstancias */  
}
```

COLOR-INDEX: indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo permita mostrar un cierto número de colores. Sus posibles valores van desde 0, que equivale a decir todos, hasta el permitido por cada dispositivo. Su valor por defecto es 0. Además, existen dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (color-index: 0){  
/* Reglas CSS aplicables para estas circunstancias */  
}  
  
@media (min-color-index: 10){  
/* Reglas CSS aplicables cuando permite, como mínimo, 10 colores */  
}
```

```
@media (max-color-index: 256){  
  /* Reglas CSS aplicables cuando permite, como máximo, 256 colores */  
}
```

- **GRID:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo de salida utiliza un sistema basado en rejilla o es un mapa de bits. Sus posibles valores son **1** o **0** que indican si el dispositivo de salida está o no basado en rejilla, respectivamente.

```
@media (grid: 0){  
  /* Reglas CSS aplicables cuando NO está basado en rejilla */  
}
```

- **HEIGHT:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la altura del dispositivo de salida se corresponda con el valor indicado. Como sus análogas, posee dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (height: 610px){  
  /* Reglas CSS aplicables para estas circunstancias */  
}  
  
@media (min-height: 40vh){  
  /* Reglas CSS aplicables para estas circunstancias */  
}  
  
@media (max-height: 80vh){  
  /* Reglas CSS aplicables para estas circunstancias */  
}
```

- **HOVER:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo de entrada permita al usuario el desplazamiento por los elementos. Sus posibles valores son HOVER, para indicar que todos los dispositivos y mecanismos que tengan disponible esta opción deben aplicar esta consulta de medios cuando esté disponible el desplazamiento por los elementos o, NONE, para cuando NO esté disponible el desplazamiento por los elementos.

```
@media (hover: hover){  
  /* Reglas CSS aplicables cuando puede desplazarse */  
}  
  
@media (hover: none){  
  /* Reglas CSS aplicables cuando NO puede desplazarse */  
}
```

- **INVERTED-COLORS:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el agente de usuario, o el sistema operativo, tenga invertidos los colores. Sus posibles valores son INVERTED, para indicar

que se aplique cuando los colores estén invertidos y, NONE, para indicar que se aplique cuando los colores se muestran normalmente.

```
@media (inverted-colors: inverted){  
  /* Reglas CSS aplicables cuando los colores están invertidos */  
}  
  
@media (inverted-colors: none){  
  /* Reglas CSS aplicables cuando los colores NO están invertidos */  
}
```

- ▶ **MONOCHROME:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo sea monocromo. Sus posibles valores son 0, para indicar que se aplique cuando NO es monocromo, o “vacío”, para indicar que se aplique cuando sí lo es.

```
@media (monochrome: 0){  
  /* Reglas CSS aplicables cuando no es monocromo */  
}  
  
@media (monochrome){  
  /* Reglas CSS aplicables cuando es monocromo */  
}
```

- ▶ **ORIENTATION:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo tenga establecida una orientación de pantalla determinada. Sus posibles valores son PORTRAIT, para indicar que se aplique cuando el dispositivo está en posición vertical, lo que equivale a decir, cuando la anchura es menor que la altura y, LANDSCAPE para indicar que se aplique cuando el dispositivo está en horizontal, es decir, el caso contrario.

```
@media (orientation: landscape){  
  /* Reglas CSS aplicables cuando el dispositivo está en horizontal */  
}  
  
@media (orientation: portrait){  
  /* Reglas CSS aplicables cuando el dispositivo está en vertical */  
}
```

- ▶ **POINTER:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando el dispositivo disponga de un dispositivo señalador (o de tipo puntero) considerado primario y tenga una precisión determinada. Sus posibles valores son FINE, para indicar que el dispositivo señalador es de alta precisión, COARSE, que indica que el dispositivo señalador es de precisión limitada o, NONE, que indica que no hay dispositivo señalador.

```
@media (pointer: coarse){  
  /* Reglas CSS aplicables cuando el dispositivo es de baja precisión */  
}
```

```
@media (pointer: fine){  
  /* Reglas CSS aplicables cuando el dispositivo es de alta precisión */  
}  
  
@media (pointer: none){  
  /* Reglas CSS aplicables cuando no hay dispositivo señalador */  
}
```

- **RESOLUTION:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la densidad en píxeles del dispositivo se corresponda con el valor indicado. Como sus análogas, posee dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (resolution: 100dpi){  
  /* Reglas CSS aplicables cuando la densidad es 100 píxeles */  
}  
  
@media (min-resolution: 72dpi){  
  /* Reglas CSS aplicables cuando la densidad es, como mínimo, 72 píxeles */  
}  
  
@media (max-resolution: 300dpi){  
  /* Reglas CSS aplicables cuando la densidad es, como mínimo, 300 píxeles */  
}
```

- **WIDTH:** indica que las reglas contenidas en la consulta de medios deberán aplicarse cuando la anchura del dispositivo de salida se corresponda con el valor indicado. Como sus análogas, posee dos variaciones que permiten controlar los valores máximo y mínimo a través de los prefijos MAX- y MIN-.

```
@media (width: 32vw){  
  /* Reglas CSS aplicables para estas circunstancias */  
}  
  
@media (min-width: 64vw){  
  /* Reglas CSS aplicables para estas circunstancias */  
}  
  
@media (max-width: 96vw){  
  /* Reglas CSS aplicables para estas circunstancias */  
}
```

4.7 FUNCIONES

A continuación, se muestra una descripción, más o menos detallada, de las funciones CSS más frecuentemente utilizadas en páginas web, a excepción de las funciones de filtro y transformaciones que se verán en un capítulo dedicado más adelante.

4.7.1 Funciones de pseudo-elementos

4.7.1.1 FUNCIÓN ATTR

La función ATTR es la abreviatura de ATTRIBUTE y tiene como objetivo devolver el valor descrito por el atributo indicado. Es muy frecuente utilizarlo para mostrar datos guardados en atributos personalizados DATA, sin embargo, puede contener cualquier atributo que se haya definido en el elemento HTML. La forma de usarlo es a través de la propiedad CONTENT de los pseudo-elementos ::BEFORE y ::AFTER.

```
div::before { content: attr(data-value); }
```

NOTA

Aunque esta función resulta ser experimental y sólo es posible utilizarla en la propiedad CONTENT de los pseudo-elementos ::BEFORE y ::AFTER, con el tiempo, es posible que pueda llegar a ofrecer muchas más funcionalidades y/o posibilidades.

4.7.1.2 FUNCIÓN COUNTER

La función COUNTER permite recuperar el valor actual de un contador CSS. Aunque los contadores CSS pueden considerarse variables CSS, su manipulación se realiza de forma muy distinta. Esta manipulación se realiza a través de las propiedades COUNTER-RESET y COUNTER-INCREMENT, las cuales permiten reiniciar una variable e incrementar su valor, respectivamente.

```
ul.falso-ol { counter-reset: indice; }
ul.falso-ol li { counter-increment: indice; }
```

Los contadores CSS pueden ser muy útiles cuando se desea representar una lista de objetos que utilizan una estructura que no está basada en listas de HTML. Por ejemplo, podría darse el caso de que se quisiera presentar un listado construido a partir de elementos gramáticos ARTICLE en donde, cada uno de ellos tuviese definido una imagen, un título, un texto introductorio y un enlace.

La forma de utilizar la función COUNTER es la siguiente:

```
ul.falso-ol li::before { content: counter(indice); }
```

4.7.2 Funciones de cálculo

4.7.2.1 FUNCIÓN CALC

El término CALC es la abreviatura de CALCULATE y tiene como objetivo realizar operaciones matemáticas como sumas, restas, multiplicaciones y divisiones.

Puede resultar muy interesante en aquellas ocasiones en donde el control del espacio disponible es muy complejo o cuando se desea que tenga una proporción determinada. No obstante, el segundo operando siempre debe ser en píxeles, em o rem.

Entre las peculiaridades que presenta esta función, cabe destacar que, todos sus operandos deben llevar asociada una unidad de medida junto al valor y que deben estar separados del operando mediante un espacio.

Ejemplos:

```
div { width: calc(100% - 20px); }
aside { right: calc(5% - 0.5em); }
nav { border-width: calc(100% - 2rem); }
```

4.7.3 Funciones gráficas

4.7.3.1 FUNCIÓN LINEAR-GRADIENT

La función LINEAR-GRADIENT tiene como objetivo crear transiciones suaves y progresivas a lo largo de una línea que viene definida a través de una dirección o ángulo entre dos o más colores separados por coma. Las transiciones lineales, también conocidas como degradados o gradiéntes lineales, pueden ser utilizadas en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

Para establecer los posibles valores de ángulo o dirección, CSS dispone de varias posibilidades. Una de ellas es especificar el lado o esquina desde donde empezará la transición. Esto es posible realizarlo a través de la palabra clave TO, seguida de una de las palabras clave LEFT, TOP, RIGHT o BOTTOM.

```
div { background-image: linear-gradient(to right, black, transparent); }
```

Otra de las formas se definir la dirección o ángulo es a través de una de las unidades de medida de ángulos que maneja CSS. Esto es:

- **DEG:** unidad de medida que representa un valor en grados. Sus posibles valores van de 0 a 360 con cualquier valor decimal.
- **GRAD:** unidad de medida que representa un valor en grados centesimales. Sus posibles valores van de 0 a 400 con cualquier valor decimal.
- **RAD:** unidad de medida que representa un valor en radianes. Sus posibles valores van desde 0 a 2π (aproximadamente 6.283184), o múltiplos del mismo.
- **TURN:** unidad de medida que representa un valor en número de vueltas. Sus posibles valores van de 0 a 1 con cualquier valor decimal.

EQUIVALENCIA	DEG	GRAD	TURN	RAD
	0	0	0	0
	90	100	0.25	1.5708
	180	200	0.50	3.1416
	270	300	0.75	4.7124

Si nos fijamos en la tabla anterior, podremos ver que todas las unidades de medida avanzan en el sentido de las agujas de un reloj. Por tanto, si lo que se desea es avanzar en sentido contrario, los valores deberán ser negativos.

```
div { border-image: linear-gradient(45deg, black, transparent); }
div { border-image: linear-gradient(50grad, black, transparent); }
div { border-image: linear-gradient(0.125turn, black, transparent); }
div { border-image: linear-gradient(0.7854rad, black, transparent); }
```

Por último, sólo destacar que el parámetro de dirección o ángulo es opcional y que, de omitirse, el degradado será vertical de arriba hacia abajo.

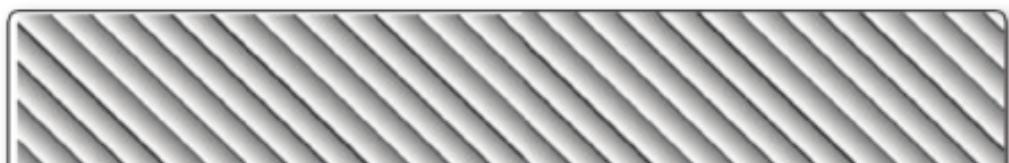
4.7.3.2 FUNCIÓN REPEATING-LINEAR-GRADIENT

La función REPEATING-LINEAR-GRADIENT tiene como objetivo crear transiciones repetitivas de tipo lineal entre dos o más colores. Al igual que la función LINEAR-GRADIENT, este tipo de transiciones o degradados es posible utilizarlos en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

La única diferencia que añade en su sintaxis es que usa un parámetro adicional que indica el espacio o tamaño al que se debe aplicar. Estos valores pueden ser establecidos en cualquiera de las unidades de medida de longitud permitidas por CSS.

```
div {
background-image: repeating-linear-gradient(
45deg,
black 45px, transparent 47px, gray 60px);
}

/* El resultado debería ser algo como: */
```



4.7.3.3 FUNCIÓN RADIAL-GRADIENT

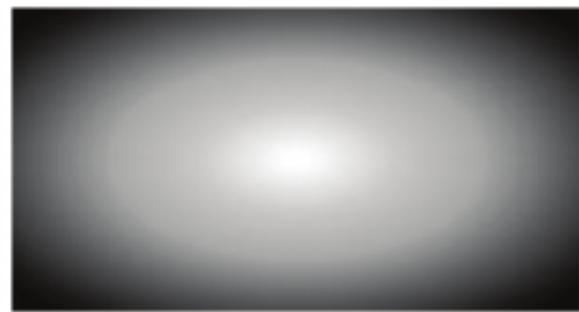
La función RADIAL-GRADIENT tiene como objetivo crear transiciones suaves y progresivas circulares o elípticas, a partir de un centro, una posición y un contorno, entre dos o más colores. Este tipo de transiciones o degradados es posible utilizarlos en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

Los degradados radiales crean, por tanto, una transición que empieza en un punto central situado en una posición concreta y a lo largo y ancho del elemento contenedor. Al igual que sucede con su variante LINEAR-GRADIENT, los valores se separan a través de comas.

Las transiciones radiales se alimentan de tres parámetros, aunque no dispone de límite. La forma y posición, color y tamaño para el punto de inicio y color y tamaño para el punto de parada. Los siguientes parámetros serán los siguientes puntos de parada.

Si, en el parámetro primer parámetro, se omite la forma y posición, la transición que se asignará al elemento será en base a las proporciones del elemento contenedor, es decir, si el elemento es cuadrado, el degradado será circular, pero si el elemento tiene forma rectangular, el degradado será elíptico.

```
div { background: radial-gradient(  
    #FFFFFF 5px, #CCCCCC 50px,  
    #AAAAAAA 100px, #000000 200px); }  
  
/* El resultado debería ser algo como: */
```

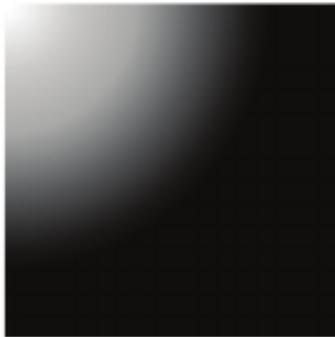


Si, en el parámetro primer parámetro, se omite sólo la posición, la transición que se asignará al elemento será la que se defina por parámetro, es decir, CIRCLE, ELLIPSE, CLOSEST-CORNER, FARTHEST-CORNER, CLOSEST-SIDE o FARTHEST-SIDE, pero la transición o degradado comenzará en el punto central del elemento contenedor.

Si, en el parámetro primer parámetro, se especifican tanto la forma, como la posición, la transición que se asignará al elemento será la que se defina por su parámetro forma, es decir, CIRCLE, ELLIPSE, CLOSEST-CORNER, FARTHEST-CORNER, CLOSEST-SIDE o FARTHEST-SIDE, y empezando en el punto indicado por la palabra clave AT.

```
div { background: radial-gradient(  
    ellipse at 0 0,  
    #FFFFFF 5px,
```

```
#CCCCCC 50px,  
#AAAAAA 100px,  
#000000 200px); }  
  
/* El resultado debería ser algo como: */
```



```
div { background: radial-gradient(  
ellipse farthest-corner at 200px 40px,  
#FFFFFF 5px,  
#CCCCCC 50px,  
#AAAAAA 100px,  
#000000 200px); }  
  
/* El resultado debería ser algo como: */
```



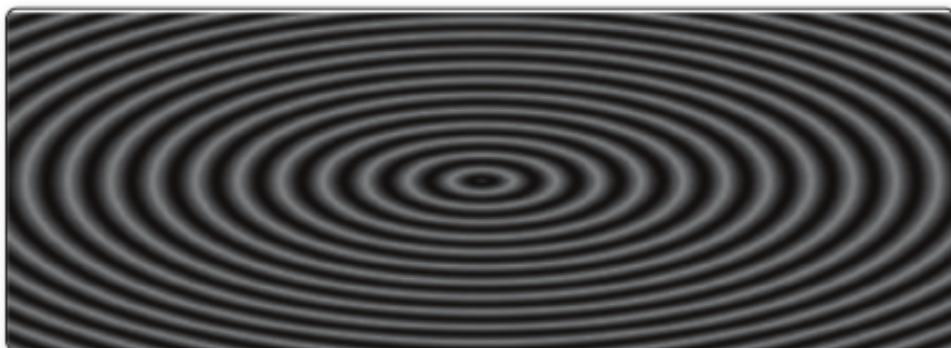
4.7.3.4 FUNCIÓN REPEATING-RADIAL-GRADIENT

La función REPEATING-LINEAR-GRADIENT tiene como objetivo crear transiciones repetitivas de tipo radial entre dos o más colores. Al igual que la función RADIAL-GRADIENT, este tipo de transiciones o degradados es posible utilizarlos en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

La única diferencia en sus sintaxis es que los degradados se van acumulando unos encima de otros en función de sus valores de tamaño, los cuales pueden ser establecidos en cualquiera de las unidades de medida de longitud permitidas por CSS.

```
div {  
background-image: repeating-radial-gradient(  
closest-side,
```

```
#000000 5px, #888888 15px, #888000 25px);  
}  
  
/* El resultado debería ser algo como: */
```

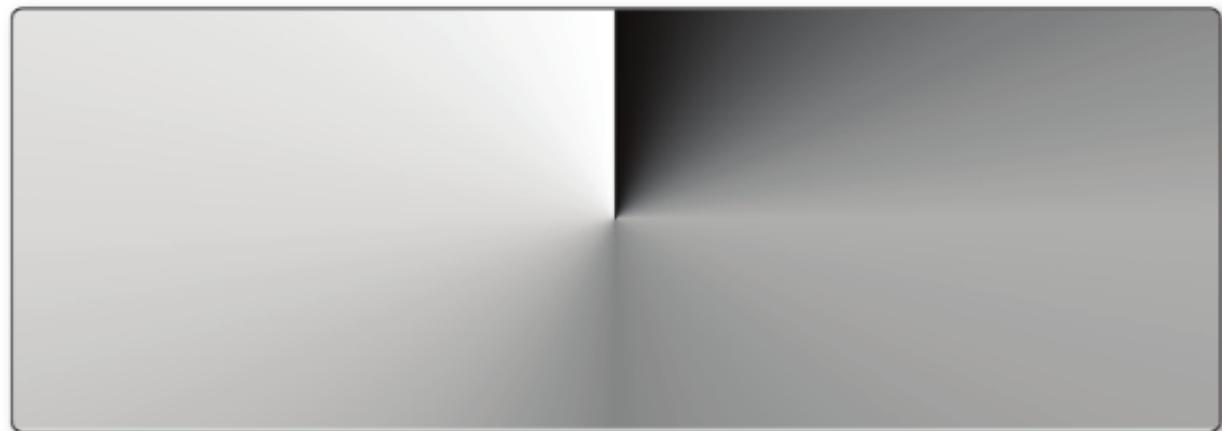


4.7.3.5 FUNCIÓN CONIC-GRADIENT

La función CONIC-GRADIENT tiene como objetivo crear transiciones de tipo cónico entre dos o más colores. Al igual que sus análogas, este tipo de transiciones es posible utilizarla en las propiedades BACKGROUND, BACKGROUND-IMAGE, BORDER-IMAGE y LIST-STYLE-IMAGE.

Su sintaxis es idéntica a LINEAR-GRADIENT y su aplicación idéntica a RADIAL-GRADIENT, con la diferencia de que la transición se aplica con respecto a un cono.

```
div {  
background: conic-gradient(black, darkgray, gray, lightgray, white);  
}  
  
/* El resultado debería ser algo como: */
```



```
div { background: conic-gradient( black 0deg, #333 0deg 10deg, #666 10deg 20deg,  
#999 40deg 120deg, #ccc 120deg 200deg, white 400deg); }  
  
/* El resultado debería ser algo como: */
```

4.8 VARIABLES

Hasta no hace tanto, la utilización de variables era una de las limitaciones que presentaba CSS. Sin embargo, en el año 2015, eso empezó a cambiar.

La necesidad de poder definir variables viene ya desde las primeras incorporaciones web a nivel profesional. Si se piensa un poco, es muy frecuente encontrarse con valores repetidos en las hojas de estilo, pero no sólo a nivel de colores, también a otros niveles como puedan ser los márgenes, animaciones o transiciones.

Las variables CSS, más correctamente denominadas **propiedades personalizadas**, pueden ser establecidas o manipuladas por varios métodos, pero su alcance lo decide la característica de herencia.

Para definir una propiedad personalizada se debe utilizar el prefijo -- (doble guion), que le indica al agente de usuario que es una variable. La forma más frecuente de declarar ésta y otras variables es recurriendo a la pseudo-clase :ROOT:

```
:root { --background: whitesmoke; }
```

Si se define una propiedad personalizada dentro de la pseudo-clase :ROOT, podrá ser utilizado por todas las reglas CSS declaradas en el documento. Sin embargo, si se define una propiedad personalizada dentro de, por ejemplo, un selector DIV, podrá ser utilizado tanto por él y todos sus pseudo-elementos, pero no podrá ser utilizado por cualquier otro elemento.

```
div {  
background: var(--backcolor);  
--forecolor: yellow;  
--backcolor: black;  
}  
  
div::before{  
content:"Texto en amarillo";  
color: var(--backcolor);  
}
```

Como se puede apreciar en la ilustración anterior, para utilizar dichas variables o propiedades personalizadas, se debe utilizar la función VAR().

Sin embargo, como hemos dicho anteriormente, no sólo es posible realizar la declaración de variables CSS a través de CSS. Esto se debe a que, esencialmente, las variables CSS se introducen el DOM del documento pudiendo ser accedidas y manipuladas a través de JavaScript. Esto es posible hacerlo mediante el uso del método SETPROPERTY perteneciente a la interfaz STYLE, que veremos en otro capítulo.

```
document.documentElement.style.setProperty('--background', 'whitesmoke');
```

Esto puede ser útil cuando se requiere manipular en tiempo real las variables para realizar, por ejemplo, estilos personalizados para cada usuario. Sólo por entender mejor esta casuística, imaginemos que tenemos una aplicación web en la que se desea que cada usuario pueda definir su tamaño de letra, colores y tipo de fuente, entre otros valores.

En un principio, declararíamos unos estilos por defecto, por si el usuario no tiene definida ninguna personalización. En concreto algo como:

```
:root {  
  --backcolor: #f0f0f0;  
  --forecolor: #003366;  
  --fontFamily: Arial, sans-serif;  
  --fontSize: 15px;  
}
```

Luego, podríamos realizar un pequeño programa en JavaScript que nos permitiese recuperar la configuración personalizada del usuario y sobrescribirla si fuese necesario.

```
var http = new XMLHttpRequest()  
  
http.open("GET", './getCustomPersonalization.php')  
http.onreadystatechange = function(){  
  if(this.readyState == 4 && this.status == 200){  
    var resultado = JSON.parse(this.responseText)  
    override(resultado);  
  }  
}  
  
http.send();  
  
function override(customJSON){  
  var style = document.documentElement.style;  
  for(key in customJSON){  
    style.setProperty('--' + key, customJSON[key]);  
  }  
}
```

4.9 PRACTICA Y JUEGA

Juego: Hex Invaders



Este es una versión de un viejo juego clásico de arcade que ha sido reconvertido a un software de aprendizaje y tiene como objetivo destruir al alienígena que posee el color mostrado. Se puede acceder desde la dirección <http://www.hexinvaders.com/>.

Juego: CSS Dinner



Aunque no usa HTML como tal, este juego puede ser de gran utilidad para aprender el uso de selectores CSS a través de elementos de marcado. El juego dispone de 32 niveles explicados detalladamente en inglés con una dificultad creciente. Se puede acceder desde la dirección <https://flukeout.github.io/>.

Juego: CSS Speedrun



Se trata de escribir, lo más rápido posible, los selectores CSS específicos que se solicitan en pantalla para los elementos resaltados. El juego dispone de 10 niveles a contrarreloj y, únicamente se encuentra disponible en inglés. Se puede acceder desde la dirección <https://css-speedrun.netlify.app/>.

Maquetación CSS – Parte 1

Corrige el CSS solicitado y juega a cambiar el HTML para realizar tu propia personalización.

<https://codepen.io/pefc/pen/LYJxrwN>

Código QR

