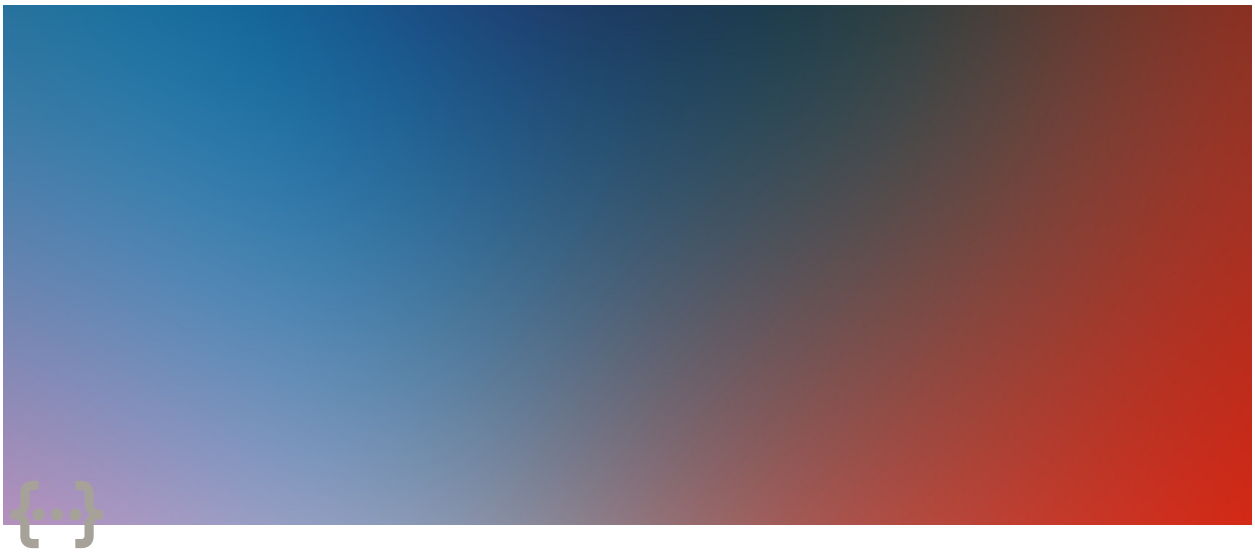


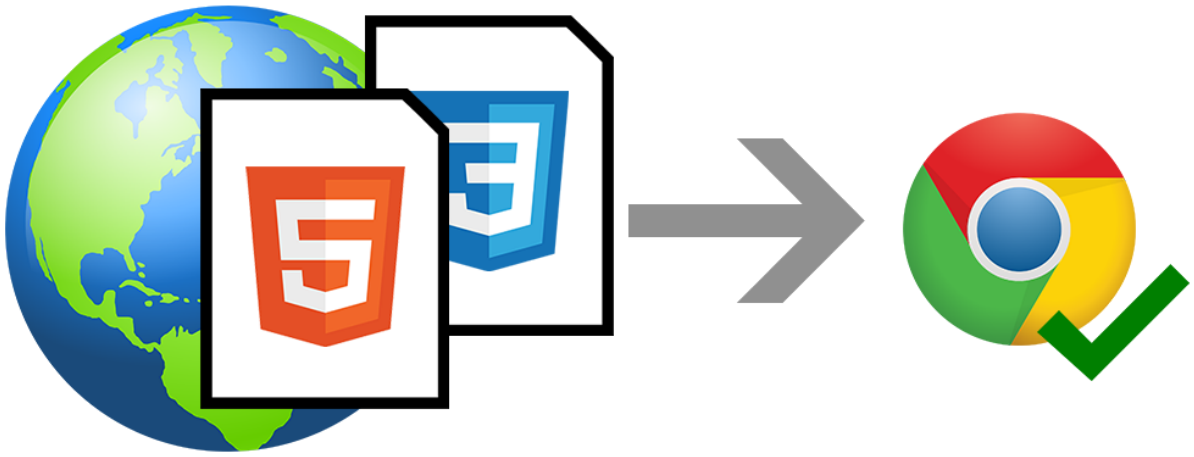
CSS

Owner	David Cazorla
Tags	CSS

Que es CSS?

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

En resumen, se trata de un «idioma» o lenguaje, como podría ser el inglés o el alemán, que los navegadores web como **Chrome** o **Firefox** conocen y pueden entender. Nuestro objetivo como diseñadores y programadores web es precisamente ese: **aprender el idioma** e **indicarle al navegador lo que debe hacer**.



¿Qué significa CSS?

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

Se le denomina estilos **en cascada** porque se lee, procesa y aplica el código desde arriba hacia abajo (***siguiendo patrones como herencia o cascada que trataremos más adelante***) y en el caso de existir ambigüedad (***código que se contradice***), se siguen una serie de normas para resolver dicha ambigüedad.

> Al empezar, siempre generalizamos mencionando que tenemos varios documentos HTML, pero sólo un documento CSS. En cada documento HTML enlazamos ese **único documento CSS**, de modo que si hacemos cambios en él, afecta a todos los documentos HTML relacionados. Esto es mucho más práctico que tener el CSS en cada uno de esos documentos, y tener que cambiarlos en cada uno de ellos.

¿Por qué se usa CSS?

La idea de CSS es la de utilizar el concepto de **separación de presentación y contenido**. Este concepto se basa en que, como programadores, lo ideal es separar claramente el código que escribimos. ¿Por qué? Porque con el tiempo, esto hará que el código sea más fácil de modificar y mantener.

La idea es la siguiente:

- Los **documentos HTML** (***contenido***) incluirán sólo información y datos, todo lo relativo a la información a transmitir.
- Los **documentos CSS** (***presentación***) inclurán sólo los aspectos relacionados con el estilo (diseño, colores, formas, etc...).

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

- Si necesitamos hacer **modificaciones visuales**, lo haremos en **un sólo lugar** y se aplica a todo el sitio.
- Se reduce la **duplicación de estilos** en diferentes lugares. Es más fácil de organizar y hacer cambios.
- La información a transmitir es considerablemente menor (***las páginas se descargan más rápido***).
- Es más fácil crear **versiones diferentes** para otros dispositivos: tablets, smartphones, etc...

Como usar CSS?

Ya tenemos claro que nuestros **documentos web** van a tener código HTML por un lado y código CSS por otro. Pero aún no sabemos como se relacionan entre sí o como se aplican. Antes de comenzar a trabajar con **CSS** hay que conocer las diferentes formas que existen para incluir estilos en nuestros **documentos HTML**.

Formas de enlazar CSS

En principio, tenemos **tres** formas diferentes de hacerlo, siendo la primera la más común y la última la menos habitual:

Mediante...	Descripción
Etiqueta `<link rel="stylesheet">`	**Archivo CSS externo**: El código se escribe en un archivo `.css` a parte.
Etiqueta `<style>`	**Bloque de estilos**: El código se escribe en una etiqueta `<style>` en el documento HTML.
Atributo HTML `style="..."`	**Estilos en línea**: El código se escribe en un atributo HTML `style` en una etiqueta.

Veamos las ventajas e inconvenientes de cada una de ellas detalladamente:

Archivo CSS externo

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

Veamos el código de nuestro `index.html`:

HTML

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

De esta forma, los navegadores sabrán que deben aplicar los estilos que se encuentren en el archivo `index.css` que está junto al documento `index.html` actual. Se aconseja escribir esta línea lo antes posible (***sobre todo, antes de los scripts***), obligando así al navegador a aplicar los estilos cuanto antes y eliminar la **falsa percepción visual** de que la página está en blanco y no ha sido cargada por completo.

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

> Antiguamente se utiliza un atributo type="text/css" que ya no es necesario en HTML5. Se puede indicar para mantener retrocompatibilidad con navegadores muy antiguos, pero actualmente se puede omitir de forma segura.

Bloque de estilos

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

HTML

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

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

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

Estilos en línea

Por último, la tercera forma de aplicar estilos en un documento HTML es el conocido como **estilos en línea**. Se trata de hacerlo directamente, a través del atributo `style` de la propia etiqueta donde queramos aplicar el estilo, colocando ahí las propiedades CSS (***que pueden separarse por*** `;`):

HTML

```
<p>¡Hola <span style="color: red; padding: 8px">amigo lector</span>!</p>
```

De la misma forma que en el método anterior, con la etiqueta `<style>`, se recomienda no utilizar este método salvo en casos muy específicos y justificados, ya que los estilos se asocian a la etiqueta HTML en cuestión y no pueden reutilizarse. Es por eso, que se suele considerar una mala práctica por muchos diseñadores cuando la sobreutilizas (***sin una razón de peso***).

Sin embargo, es una excelente forma de inyectar o incluir variables CSS en una etiqueta y sus etiquetas hijas.

> Consejo: Si quieres comenzar a hacer pruebas rápidas con HTML, CSS y Javascript puedes utilizar CodePen, una plataforma web que te permite

> crear contenido HTML, CSS y Javascript, previsualizando en tiempo real.

Estructura de CSS

Los **documentos CSS** son archivos de texto (***igual que los documentos HTML***), sólo que en este caso tienen extensión `.css` en lugar de `.html`. En el código de un documento `.css` se escriben una serie de órdenes y el cliente (***navegador***) las interpreta y aplica a los documentos HTML.

En este artículo vamos a ver como es la sintaxis de un archivo `.css` y como se debe escribir el código CSS, a la vez que ponemos en práctica lo aprendido en los artículos anteriores.

Relación del HTML con CSS

En primer lugar, recuerda que debemos tener el documento `.css` enlazado desde nuestro documento `.html`, preferiblemente desde una etiqueta `<link rel="stylesheet">`. En su atributo `href` colocaremos el nombre del documento `.css` que contiene los estilos:

HTML

```
<!DOCTYPE html><html>
<head>
<title>Título de página</title>
<link rel="stylesheet" href="index.css" />
</head>
<body>
<div class="page">
<h1>Título de la página.</h1>
<p>Me encanta Manz.dev. Lo veo en Twitch o Youtube todos los días.</p>
</div>
</body>
</html>
```

Bien, ya tenemos nuestro archivo `.html` conectado con nuestro archivo `.css`. Ahora todo el código CSS que coloquemos en nuestro `index.css` se aplicará al HTML y se verá reflejado en el navegador.

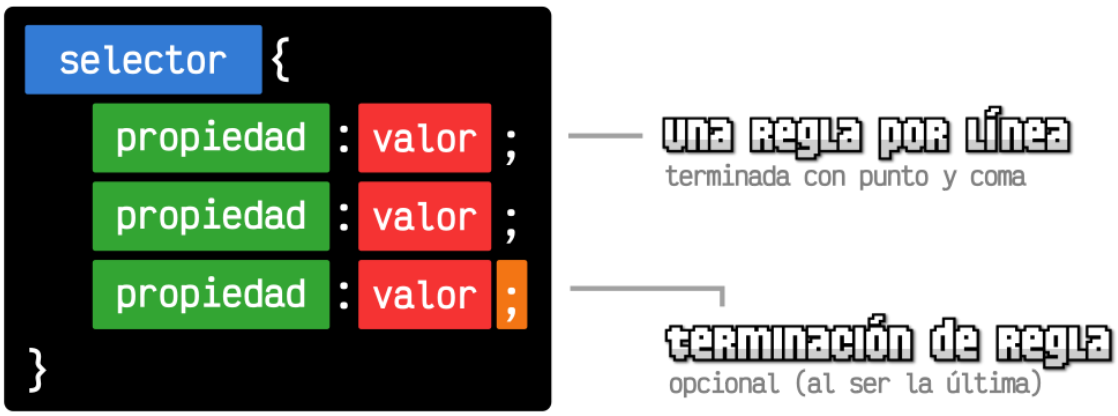
Sintaxis CSS

Vamos a centrarnos ahora en el contenido de nuestro documento `index.css`. El **código CSS** de dicho documento, se basa en una serie de conceptos que debemos tener claros antes de continuar. Demos un repaso a cada uno de ellos:

Concepto	Descripción

Selector	Elemento o elementos del documento que vamos a **seleccionar** para aplicarle un estilo concreto.
Propiedad	**Característica principal** que vamos a definir con el valor indicado a continuación.
Valor	Cada **propiedad** tiene una serie de valores concretos asignables, con los que tendrá uno u otro comportamiento.
Comentario	Fragmento de texto entre `/*` y `*/` con anotaciones o aclaraciones para el desarrollador (el navegador los ignorará).
Regla	Par de **propiedad** y su respectivo **valor** asociado.

Así pues, tras aprender estos conceptos, la **estructura** de un documento `.css` se basaría en organizar el código siguiendo el formato del siguiente bloque:



Vamos a verlo con un ejemplo para afianzar conceptos:

HTML

CSS

DEMO

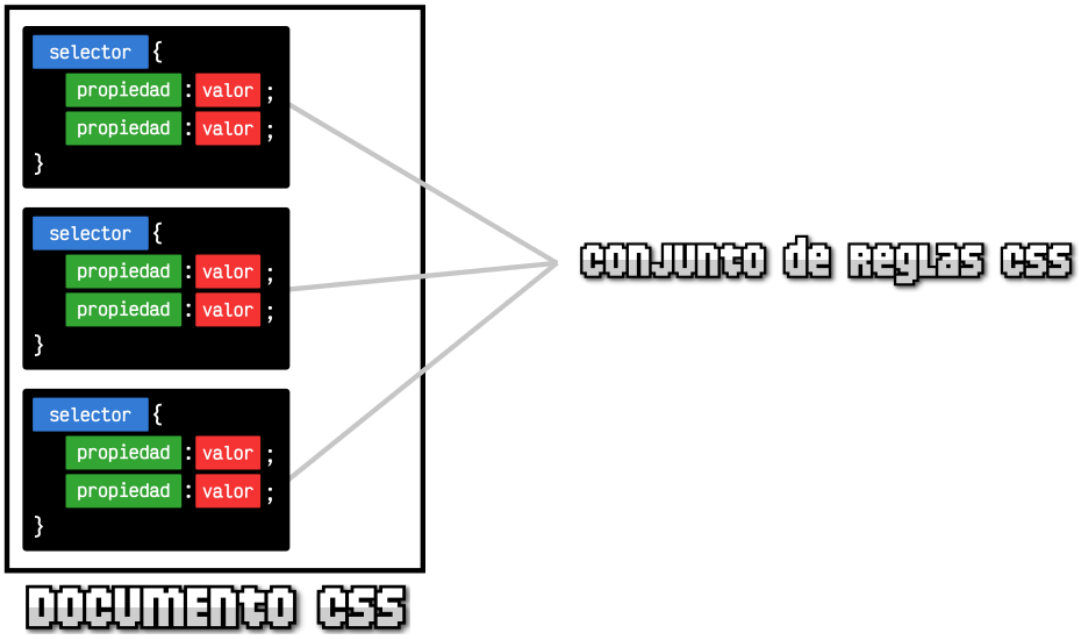
```
<div class="page"><h1>Título de la página.</h1><p>Me encanta Manz.dev. Lo veo en Twitch o Youtube todos los días.</p></div>
```

En este caso, estamos seleccionando todas las etiquetas `<p>` del documento HTML (***en este ejemplo es una sola, pero si existieran más se aplicaría a todas***), y les aplicaremos las reglas que contenga, en este caso una regla que define el **color de texto rojo**. Cada una de estas reglas se terminará con el carácter **punto y coma** (`;`), seguido de la siguiente regla. El último punto y coma es opcional y se puede omitir si se desea.

> Ojo: Como hemos visto, se pueden incluir comentarios entre los caracteres /* y */, los cuales serán ignorados por el navegador. Estos suelen servir para añadir notas o aclaraciones dirigidas a humanos.

Documento CSS

Sin embargo, esto es sólo un ejemplo muy sencillo. Los **documentos CSS** se forman por cientos de reglas de este estilo, con diferentes selectores, propiedades y valores. Habitualmente, cada bloque declarado suele incluir múltiples reglas (***y no sólo una como en el ejemplo anterior***):



Ten en cuenta que, a medida que escribimos **código CSS**, este se va haciendo **más grande** y más difícil de controlar y mantener. Para intentar evitar perder el control del mismo, hay que seguir una serie de buenas prácticas al escribir código, que aunque no son necesarias para que funcione en el navegador, los desarrolladores consideramos obligatorias para que sea más fácil de leer:

- Escribe **una regla por línea**. Será mucho más fácil de leer y modificar.
- Usa la **indentación**. Tras escribir el carácter `{` al empezar un bloque de reglas, las propiedades se separan hacia la derecha. Esto hace que sea más fácil de leer y se considera una buena práctica de desarrollo y de programación en general.
- El **último punto y coma** de un bloque de reglas es opcional. Sin embargo, se suele aconsejar escribirlo para mantener una coherencia y evitar problemas o descuidos al modificar posteriormente el código.

Como puedes ver, estos consejos mejoran sustancialmente la legibilidad del código. Se consideran prácticas de cumplimiento obligatorio para ayudar a entender más rápidamente el código ajeno (***o incluso el nuestro***) sin necesidad de invertir mucho tiempo.

Más adelante, en un capítulo dedicado expresamente a ello, veremos que la estructura CSS puede ser más compleja, pero de momento trabajaremos con este esquema simplificado.

Navegadores web

Los **navegadores web** (***también llamados clientes***) son esos programas que utilizamos para acceder a Internet y visualizar páginas en nuestros dispositivos. Todos los usuarios conocen al menos uno o varios navegadores web, aunque sea los más populares como **Google Chrome** o **Mozilla Firefox**. Sin embargo, existen muchos más. Para ser un buen diseñador o desarrollador web es recomendable conocer bien el ecosistema de navegadores existente y sus principales características, que no son pocas.

Ecosistema de navegadores

En un mundo ideal, todas las páginas webs se verían correctamente y de la misma forma en todos los navegadores web disponibles, sin embargo, y una de las cosas que más llama la atención del diseño web cuando estamos empezando, es que no sólo debemos construir una web correctamente, sino que además **debemos ser conscientes de los navegadores más utilizados**, así como de sus carencias y virtudes.

En un principio, el consorcio **W3C** se encarga de definir unas especificaciones y «normas» de recomendación, para que posteriormente, las compañías desarrolladoras de navegadores web las sigan y puedan crear un navegador correctamente. Pero como no estamos en un mundo perfecto (***y el tiempo es un recurso limitado***), dichas compañías establecen prioridades, desarrollan características antes que otras, e incluso algunas características deciden no implementarlas por razones específicas o internas.

Las compañías más comprometidas con sus navegadores web, tienen a disposición de los diseñadores, programadores y entusiastas, una especie de diario cronológico, donde mencionan su hoja de ruta con las características que van implementando, descartando o sus planes de futuro, así como información adicional sobre el tema en cuestión:

Responsables	**Producto**	**Página de desarrollo**
(Proyecto open source)	Motor Webkit	Webkit Feature Status
Fundación Mozilla	Navegador Mozilla Firefox	Mozilla Standards Positions
Google	Navegador Google Chrome	Chrome Platform Status

También podemos ver los últimos cambios en WebStatus, una especie de resumen de novedades.

Historia de los navegadores

Si echamos un vistazo atrás, la historia de los navegadores ha variado muchísimo. Quizás, el cambio más importante en los últimos 10 años ha sido el reemplazo de **Internet Explorer**, como navegador más popular, a **Google Chrome**. Antiguamente, **Internet Explorer** fue un navegador que se había estancado y no implementaba nuevas características y funcionalidades, al contrario que sus competidores. Pero además, para empeorar la situación, era el navegador más utilizado

por los usuarios, debido al liderazgo de Windows como sistema operativo. Esto impedía que las nuevas tecnologías webs se adoptaran y frenaba su avance. Por suerte, esto ha ido cambiando a lo largo de los años y la situación hoy en día es bastante diferente.

A continuación se puede ver la evolución de los navegadores más populares durante esta última década (***desde 2009 hasta 2016***). Vemos que los navegadores más perjudicados son Internet Explorer y Mozilla Firefox, mientras que Chrome ha experimentado un incremento muy grande. Safari también ha experimentado un ligero incremento, probablemente debido al éxito de dispositivos como iPhone o iPad.

Como toda estadística, debe ser tomada con precaución porque existen sesgos en sus datos. Esta estadística ha sido extraída de Global StatCounter. También puedes echar un vistazo a algunas estadísticas más en w3counter, aunque quizás la más interesante y adecuada sea CanIUse: Usage table, donde podemos encontrar los navegadores más utilizados, separado por versiones y mostrado con porcentajes.

Navegadores actuales

A continuación, tenemos una lista de la rama de los **5 navegadores más populares**, que son aquellos que tienen una cuota de mercado considerable. Algunos de estos navegadores tienen varias versiones diferentes, como por ejemplo, versiones beta (***con funcionalidades aún no existentes en la versión oficial***) o versiones de desarrollador (***orientadas para el uso de programadores o diseñadores***).

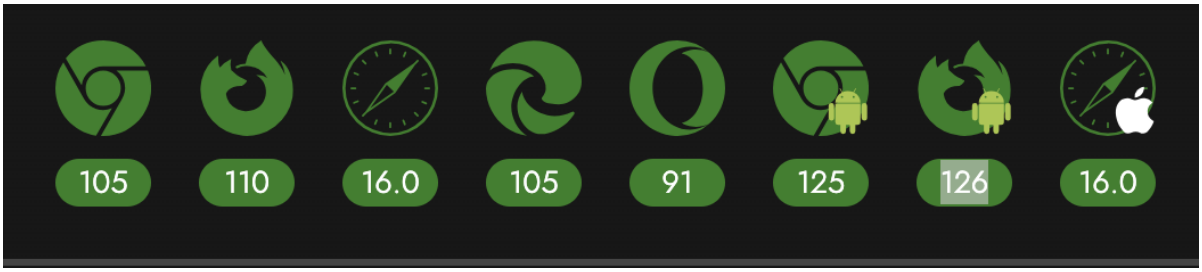
Responsables	Navegador web	Propósito	Motor	Notas	URL
Google	**Chrome**	Uso habitual	Blink		URL
Google	Chrome beta	Early-adopters	Blink		URL
Google	Canary Chrome	Desarrollador	Blink		URL
Google	Chromium	Open Source	Blink		URL
Mozilla	**Firefox**	Uso habitual	Quantum		URL
Mozilla	Firefox beta	Early-adopters	Quantum		URL
Mozilla	Firefox Dev Edition	Desarrollador	Quantum		URL
Mozilla	Firefox Nightly	Desarrollador	Quantum		URL
Microsoft	**Edge**	Uso habitual	Blink		URL
Opera	**Opera**	Uso habitual	Blink		URL
Opera	Opera beta	Early-adopters	Blink		URL
Opera	Opera GX	Streamers/Gaming	Blink		URL
Opera	Opera Neon	Early-adopters	Blink		URL
Apple	**Safari**	Uso habitual	Webkit	Sólo Apple	URL

Apple	Safari Tech Preview	Desarrollador	Webkit	Sólo Apple	URL

Versiones de los navegadores

Es también muy importante conocer la versión del navegador que utiliza la mayoría de nuestro público (***datos que se pueden obtener con herramientas como Google Analytics, por ejemplo***), ya que de una versión a otra se añaden nuevas características y funcionalidades, de las cuales los usuarios de las versiones anteriores no podrán disfrutar.

En la herramienta CanIUse se muestra, a lo largo de las columnas de cada navegador, si las funcionalidades están implementadas en la versión concreta del mismo, o cuando empezarán a funcionar. En esta documentación, encontrarás un widget similar al siguiente donde podrás comprobar dicha compatibilidad desde esta misma página:



Es por tanto, lógico pensar, que si tenemos un alto porcentaje de usuarios que utilizan una versión de un navegador que no soporta la funcionalidad que queremos utilizar, haya que buscar alternativas o abstenerse a utilizarla hasta que ese porcentaje se reduzca.

> Por suerte, desde hace ya bastante tiempo los navegadores han comenzado a implementar una estrategia de actualización silenciosa (en inglés: evergreen browser), con la cuál consigues que el usuario con conexión a Internet tenga siempre el navegador actualizado a su última versión. Esto es así, ya que el grueso de los usuarios no suele actualizar manualmente la versión de su navegador, y esto provocaba que existiera una gran cuota de usuarios con navegadores sin actualizar.

Otros navegadores

A continuación, tenemos una lista de otros navegadores menores, que no superan una cuota de mercado a nivel global de un 1%, pero que pueden ser interesantes en el futuro, para casos particulares o podrían experimentar un aumento de su cuota en los próximos años:

Responsables	Navegador web	Propósito	Motor	Basado en	URL
Tor Project	Tor	Navegación anónima	Gecko	Firefox	URL
Vivaldi Tech	Vivaldi	Early-adopters	Blink		URL
Brave Soft	Brave	Privacidad	Blink		URL
Yandex	Yandex Browser	Uso habitual	Blink		URL
David Rosca	Falkon	Uso habitual	Qt WebEngine		URL
Maxthon Int	Maxthon	Uso habitual	Trident/Webkit		URL
Fenrir Inc	Sleipnir	Uso habitual	Blink		URL
SM Project	SeaMonkey	Uso habitual	Gecko	Mozilla AS	URL
Chris Dywan	Midori	Uso habitual	Webkit		URL
Comodo Group	Comodo Dragon	Uso habitual	Blink	Chromium	URL
Browsh	Browsh	Navegador de texto	Gecko	Firefox	URL
Microsoft	Internet Explorer	Descontinuado	Trident		URL

Existen muchos más navegadores, esto sólo es una lista de los que he considerado más relevantes.

Navegadores de texto

Los siguientes navegadores son navegadores para terminales de texto puro, útiles en determinados ámbitos o para tareas específicas, como por ejemplo, comprobar como se ve una web en dispositivos que tienen funcionalidades limitadas:

Responsables	Navegador web	Propósito	Motor	Basado en	URL
M. Patocka	Links	Navegador de texto	-		URL
Thomas Dickey	Lynx	Navegador de texto	-		URL
Akinori Ito	w3m	Navegador de texto	-		URL
P. Baudis	Elinks	Navegador de texto	-		URL

Herencia

La herencia en CSS es un mecanismo mediante el cual ciertos valores de las propiedades de un elemento se transfieren automáticamente a sus elementos hijos. Esto permite que los estilos se apliquen de manera consistente y eficiente sin necesidad de especificarlos repetidamente para cada elemento.

Características Clave:

- **Propiedades Heredadas**: Algunas propiedades, como `color`, `font-family`, y `line-height`, son heredadas por defecto.
- **Propiedades No Heredadas**: Otras propiedades, como `margin`, `padding`, y `border`, no se heredan por defecto.
- **Palabra Clave** `inherit`: Se puede usar explícitamente para forzar la herencia de una propiedad, por ejemplo, `color: inherit;`.
- **Control de Herencia**: CSS también proporciona palabras clave como `initial`, `unset`, y `revert` para controlar la herencia y los valores iniciales.

Ejemplo:

```
body {
    color: blue; /* Esta propiedad será heredada por todos los hijos del body */
}

p {
    font-family: 'Arial'; /* Esta propiedad también será heredada por los hijos del p */
}

div {
    margin: 20px; /* Esta propiedad no será heredada */
}
```

En este ejemplo, todos los elementos dentro del `body` tendrán el texto azul y todos los elementos `p` tendrán la fuente 'Arial'. Sin embargo, solo los elementos `div` tendrán un margen de 20px, ya que `margin` no es una propiedad heredada.

herramienta recomendada

https://codi.link

Pseudo-classes

Una **pseudoclase CSS** es una palabra clave que se añade a los selectores y que especifica un estado especial del elemento seleccionado. Por ejemplo, `:hover` aplicará un estilo cuando el usuario haga hover sobre el elemento especificado por el selector.

```
div:hover {
  background-color: #F89B4D;
}
```

Las pseudoclases, junto con los pseudoelementos, permiten aplicar un estilo a un elemento no sólo en relación con el contenido del árbol de documento, sino también en relación a factores externos como el historial del navegador (`:visited`, por ejemplo), el estado de su contenido (como `:checked` en algunos elementos de formulario), o la posición del ratón (como `:hover` que permite saber si el ratón está encima de un elemento o no).

Nota: En lugar de usar pseudoclases, `pseudo-elements` puede usarse para dar estilo a una *parte específica* de un elemento.

Sintaxis

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

Al igual que las clases, se pueden concatenar la cantidad de pseudoclases que se deseen en un selector.

Indice de las pseudo-clases estándar

- `:active`
- `:checked`
- `:default`
- `:dir()`
- `:disabled`
- `:empty`
- `:enabled`
- `:first`
- `:first-child`
- `:first-of-type`
- `:fullscreen`

- `:focus`
- `:hover`
- `:indeterminate`
- `:in-range`
- `:invalid`
- `:lang()`
- `:last-child`
- `:last-of-type`
- `:left`
- `:link`
- `:not()`
- `:nth-child()`
- `:nth-last-child()`
- `:nth-last-of-type()`
- `:nth-of-type()`
- `:only-child`
- `:only-of-type`
- `:optional`
- `:out-of-range`
- `:read-only`
- `:read-write`
- `:required`
- `:right`
- `:root`
- `:scope`
- `:target`
- `:valid`
- `:visited`

Selectores Combinados

Los selectores combinados en CSS se utilizan para aplicar estilos a elementos HTML que cumplen con múltiples criterios. Estos selectores permiten crear reglas de estilo más específicas y potentes. A continuación, se presentan algunos de los selectores combinados más comunes:

Selectores de Descendiente

Aplica estilos a los elementos que son descendientes de un elemento especificado (hijos, nietos, bisnietos, etc.).

```
div p {
  color: blue;
}
```

En este ejemplo, todos los párrafos (`<p>`) dentro de cualquier `<div>` se estilizarán con color azul.

Selectores de Hijo

Aplica estilos a los elementos que son hijos directos de un elemento especificado.

```
ul > li {
  color: green;
}
```

En este ejemplo, solo los elementos de lista (`<li>`) que son hijos directos de un `<ul>` se estilizarán con color verde.

Selectores de Hermano Adyacente

Aplica estilos al elemento que inmediatamente sigue a otro elemento.

```
h1 + p {
  margin-top: 0;
}
```

En este ejemplo, el párrafo (`<p>`) inmediatamente después de un `<h1>` tendrá el margen superior establecido en 0.

Selectores de Hermano General

Aplica estilos a todos los elementos hermanos que siguen a un elemento especificado.

```css
h1 ~ p {
  color: red;
}
```

En este ejemplo, todos los párrafos (`<p>`) que siguen a un `<h1>` (no solo el primero) se estilizarán con color rojo.

Selectores de Atributo

Aplica estilos a los elementos que tienen un atributo específico o un atributo con un valor específico.

```css
input[type="text"] {
  background-color: yellow;
}
```

En este ejemplo, todos los campos de entrada (`<input>`) de tipo texto tendrán un fondo amarillo.

Selectores Combinados de Clase e ID

Aplica estilos a los elementos que tienen una combinación específica de clases y/o IDs.

```css
#menu .active {
  font-weight: bold;
}
```

En este ejemplo, el elemento con la clase `.active` dentro del elemento con el ID `#menu` tendrá el texto en negrita.

unidades no absolutas

```
    <div class="container">
   <h1>hola</h1>
```

```
    </div>
```

```
.container{
    width: 50px;
    height: 50px;
    background-color: blue;
}
```

en relación al viewport vh, vw

- [ ] modelo de la caja estilo en linea y en bloque
- [ ] en estilos en linea los elementos no se modifican el alto y ancho
- [ ] propiedad box-sizing (conten-box, border-box)
- [ ] fixed y sticky