2023/24

U.D.3. USO DE ESTILOS.

3.1 Introducción a CSS.

ÍNDICE

1. Introducción

Cuando se trabaja con páginas web escritas en HTML y XHTML, es deseable que presenten un estilo concreto y un aspecto definido. La forma más eficiente para poder llevar esto a cabo es distinguir entre la parte en la que se sitúan los **contenidos** (**HTML**), y la parte del proyecto donde se recogen los aspectos relativos a la **apariencia**. Aquí es donde aparece **CSS**.

Las hojas de estilo en cascada **(CSS - Cascading Style Sheets) son un estándar W3C que define la presentación de los documentos Web**, es decir, el modo en el que se muestra un documento en pantalla o se suministra al usuario, ya sea por el monitor o en la pantalla del teléfono móvil.

El lenguaje **CSS se ha ido creando a lo largo del tiempo en varios niveles**. Cada nivel de CSS se ha construido sobre el anterior, generalmente añadiendo funcionalidades nuevas. En la página oficial de W3C (w3.org/Style/CSS/) se pueden consultar todas las publicaciones relacionadas con las novedades del estándar CSS en sus diferentes versiones.

El estándar CSS2 define más de 100 propiedades, cada una de ellas con su lista de valores permitidos. Por su parte, el estándar **CSS3 ya incluye más de 200 propiedades o atributos**.

El uso de CSS aporta numerosas ventajas:

- **Mayor control** en el diseño de las páginas. Se puede llegar a diseños fuera del alcance de HTML.

- **Menos trabajo**. Se puede cambiar el estilo de todo un sitio con la modificación de un único archivo. Esto es, **con un archivo CSS se pueden modificar varios archivos html**.

- **Documentos más pequeños**. Las etiquetas <font> y la gran cantidad de tablas empleadas para dar una buena apariencia a los sitios web desaparecen ahora, por lo que se ahorra código en la configuración de la presentación del sitio.

- **Documentos mucho más estructurados.** Los documentos bien estructurados son accesibles a más dispositivos y usuarios. Por tanto, facilita la visualización del sitio en dispositivos diferentes.

- **El HTML de presentación está en vías de desaparecer**. Todos los elementos y atributos de presentación de las especificaciones HTMLy XHTML fueron declarados obsoletos por el W3C.

- **Tiene buen soporte**. En este momento, casi todos los navegadores soportan casi toda la especificación CSS1 y la mayoría también las recomendaciones de nivel 2 y 2.1.

- Mejora la **accesibilidad** del documento.

- **Reduce la complejidad de su mantenimiento**, tanto si va a ser realizado por uno mismo como por terceros.

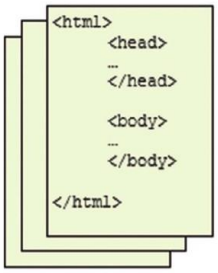
1.1 Funcionamiento

El **proceso de funcionamiento** de las hojas de estilo en cascada podemos resumirlo en tres pasos:

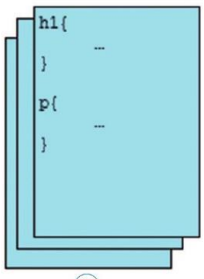
1. Hay que comenzar con un documento **HTML** (o XHTML). En teoría, el documento tendrá una estructura lógica y un significado semántico a través de los elementos HTML adecuados. Con HTML se crea la **estructura de la página web**.

2. Luego hay que escribir las **reglas de estilo** para definir el **aspecto** ideal de todos los elementos. Las reglas seleccionan el elemento en cuestión por su nombre y, a continuación, listan las propiedades (fuente, color, etc.) y los valores que se le van a aplicar.

3. Por último, hay que **vincular los estilos al documento**. Las reglas de estilo pueden reunirse en un documento independiente y aplicarse a todo el sitio, o pueden aparecer en la cabecera y aplicarse solo a ese documento.



①



②

③

El principal **inconveniente** de no utilizar los documentos CSS y hacerlo desde el propio código en HTML es que, **si se quiere hacer una modificación** en los estilos definidos, **es necesario modificar todas las páginas** que incluyen el estilo que se va a modificar. Es por ello que a la hora de escribir código CSS, se **recomienda** lo siguiente:

a) Tener **un archivo CSS con los estilos más globales** que vayan a tener todas las páginas HTML.

b) Asociar a cada archivo HTML que tenga sus características o estilos más concretos, a un **documento CSS con estilos más específicos**.

1.2 Soporte CSS en los navegadores web

No todos los navegadores ofrecen el mismo soporte a las guías de estilo. Debe conocerse el motor de cada navegador, dado que es la parte que se encarga de interpretar el código HTML y CSS. En el siguiente cuadro se muestran los soportes de las diferentes versiones de CSS en los principales navegadores que se utilizan en la actualidad.

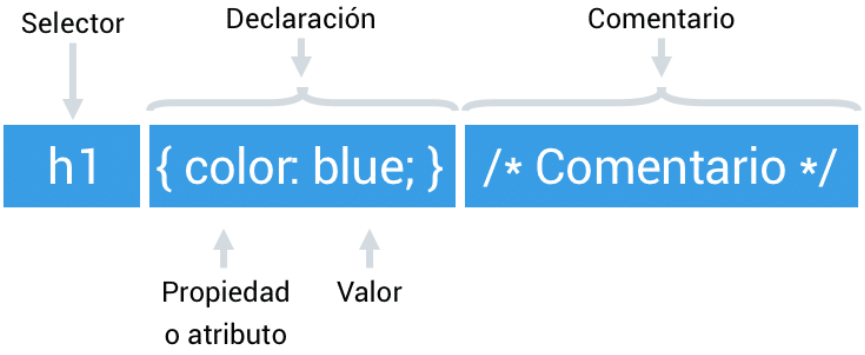
Navegador	CSS1	CSS2	CSS3	Motor
Firefox	Completo	Completo	Todos los selectores y pseudoclases, y ciertas propiedades	Gecko
Explorer	Completo	Completo	?	?
Chrome	Completo	Completo	Todos los selectores y pseudoclases, y muchas propiedades	Webkit
Safari	Completo	Completo	Todos los selectores y pseudoclases, y muchas propiedades	WebKit
Opera	Completo	Completo	Todos los selectores y pseudoclases, y muchas propiedades	Presto

https://caniuse.com/?compare=firefox+95,chrome+98,safari+15,opera+80&compareCats=CSS

Los **desarrolladores** siempre deben comprobar que sus webs se visualizan correctamente en los diferentes navegadores.

1.3 Sintaxis CSS

En CSS se utiliza la siguiente sintaxis para asignar valores a las propiedades de cada selector:



- ✓ **Selector:** indica sobre qué elemento se aplican los estilos CSS.
- ✓ **Propiedad o atributo**: indica qué característica se va a cambiar.
- ✓ **Valor:** indica el valor de la propiedad que se desea modificar.
- ✓ **Comentario:** los comentarios se escriben entre el carácter de apertura "`/*`" y el carácter de cierre "`*/`" (por ejemplo: `/* Este es un comentario */`)

1.4 CSS3

CSS3 (*Cascading Style Sheet 3*) es la tercera versión del lenguaje de hojas de estilo CSS. En este apartado se verán algunas de las novedades que presenta frente a su versión anterior, CSS 2.1.

Esta nueva actualización presenta todas las **características** de su predecesora y, además, aporta un mayor dinamismo en el diseño de la página web. En cuanto a los aspectos relativos a la programación, esta no varía, se sigue empleando la **estructura de propiedades**.

A. MÚLTIPLES IMÁGENES EN EL FONDO.

En esta nueva versión es posible poner **varias imágenes de fondo en un mismo elemento**. En el siguiente ejemplo, se puede ver como se utilizan varias imágenes para el fondo y, a continuación, de la URL de cada una de ellas indica las posiciones del background.

```
#contenidoPrincipal {
      width: 500px;
      height: 500px;
      background:
            url(imagen1.gif) bottom right no-repeat,
            url(imagen2.jpg) center center no-repeat,
            url(imagen3.png) repeat;
}
```

B. NUEVAS PROPIEDADES PARA EL FONDO DE LOS ELEMENTOS.

En CSS3 se incorporan nuevas propiedades que definen las características del fondo de los elementos. Estas son ***background-origin***, ***background-clip*** y ***background-size***.

- ***Background-origin***: especifica la posición de origen de un fondo o imagen.
- ***Background-clip***: permite definir el comportamiento de un fondo o imagen de fondo, con respecto al borde, si debe mostrarse por debajo de este o no. Esta propiedad se aplica a todas las cajas contenedoras conocidas: border-box, padding-box y content-box.
- ***Background-size***: permite definir el tamaño de las imágenes de fondo.

C. ESQUINAS REDONDEADAS.

Esta nueva propiedad permite redondear las esquinas de las áreas de la página, en concreto, a través del atributo ***border-radius***.

```
div {
      border: 2px solid #FFFFFF;
      border-radius: 9px;
}
```

D. SOMBRAS.

Esta versión incorpora una propiedad para crear sombras en los elementos, incluso en los textos. El atributo que se utiliza es ***box-shadow*** y está formado por **cuatro valores**: desplazamiento **horizontal** de la sombra, desplazamiento **vertical** de la sombra, **difuminado y color** de la sombra; estos valores aparecen siempre en el **mismo orden**.

```
div {
      box-shadow: 5px -9px 3px #000000;
}
```

E. TRANSPARENCIAS DE COLOR.

Otra de las nuevas incorporaciones en CSS3 es la posibilidad de añadir transparencia en el color; desde un color totalmente sólido hasta el transparente. La propiedad que define la transparencia se denomina ***opacity***, y toma un valor decimal **entre 0.0 y 1.0**, donde 0.0 es la máxima transparencia y **1.0 la máxima opacidad**.

```
div {
      opacity: 0.5;
}
```

F. TEXTO EN VARIAS COLUMNAS.

En CSS3 se incorpora la opción de maquetar el texto en varias columnas a través de varios atributos:

- ***columns***: define el número de columnas.
- ***column-width***: define el ancho de las columnas.
- ***column-gap***: define el espacio en blanco entre columnas.
- ***column-rule***: crea una línea entre las columnas.

```
<!DOCTYPE html>
<html>
<head>
<style>
    .newspaper {
         columns: 4;
         column-width: 100px;
         column-gap: 20px;
         column-rule: 2px solid #EEE;
        }
</style>
</head>
<body>

    <h1>The column Property</h1>
    <p>The column-width property specifies the column width:</p>

    <div class="newspaper">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad
minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut
aliquip ex ea commodo consequat.
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie
consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan
et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis
dolore te feugait nulla facilisi.

</div>

</body>
</html>
```

En pantalla se verá:

The column Property

The column-width property specifies the column width:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

G. ANIMACIONES.

Unas de las novedades más importantes es la creación de animaciones. Estas se crean directamente desde CSS3 **sin la necesidad de utilizar otros lenguajes**, además, con este tipo de implementación **ya no será necesario tener instaladores plugins**, que antes eran inevitables para la correcta visualización de las animaciones.

2. Modos de aplicación

A la hora de aplicar las reglas de estilo a un documento (X)HTML, se debe tener en cuenta que existen tres modos distintos:

- ✓ Estilos **en línea**.
- ✓ Hojas de estilos **incrustados**.
- ✓ Hojas de estilos **externas**: **vinculadas o importadas**.

2.1 Estilos en línea

Se puede añadir información de **estilo a un elemento concreto** de un documento (X)HTML, empleando el atributo ***style*** en la etiqueta del elemento. Esta operación se realiza en el propio documento. Su **sintaxis** general es:

```
<etiqueta HTML style=”declaraciones de estilo”>
```

Aunque este es un uso perfectamente válido de la información de estilo, los estilos en línea **contaminan el documento** con información de presentación.

Un **ejemplo** de este modo de aplicación sería el siguiente:

```
<p style="color:green">Párrafo de color verde.</p>
```

Debe **emplearse con moderación**, solo ocasionalmente, para ignorar reglas de nivel más alto.

De hecho, el atributo ***style*** fue depreciado en XHTML 1.1 y no aparece en otros lenguajes XML.

2.2 Hojas de estilos incrustadas en la cabecera

Un método más compacto para añadir hojas de estilo es incrustar un bloque de estilo en la **cabecera** del documento (X)HTML empleando el elemento ***style***.

El siguiente **ejemplo** muestra la sintaxis general de un documento (X)HTML donde se ha incrustado una hoja de estilo mediante el uso del elemento ***style***.

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
                color: red;
                text-align: center;
                }
        </style>

    </head>

    <body>
        <p> Parrafo 1 </p>
        <p> Parrafo 2 </p>

    </body>
</html>
```

El elemento ***style*** debe colocarse en la cabecera del documento (dentro del ***head***).

Este método **puede emplearse cuando sólo tenemos un documento web que utiliza un único estilo.**

La **desventaja** de este método es que, a la hora de realizar **cualquier cambio, se debe realizar en múltiples páginas diferentes y el código estará repetido**.

2.3 Hojas de estilos externas

Mediante hojas de estilo externas se consigue **separar el archivo de estilos del fichero HTML**. El archivo de estilos cuenta con la **extensión .css** y **se referencia desde HTML**.

Este es el **método más eficiente y más sencillo de mantener** ya que el código CSS se encuentra separado del fichero HTML.

Las hojas de estilos externas pueden ser, a su vez, **incrustadas o vinculadas**.

2.3.1 Hojas de estilo incrustadas

Se puede importar una hoja de estilo con la sentencia ***@import*** de CSS. Esta sentencia puede usarse **en un archivo CSS** o dentro del elemento ***style*** visto anteriormente.

```
<head>
    <style>
        @import url|string lista-de-mediaqueries;

    </style>

</head>
```

Hay que **tener en cuenta** que si se utiliza este método:

- Las sentencias ***@import*** deben estar al principio de la hoja de estilo, antes de cualquier otra regla.

- El orden en el que se importan las hojas de estilo importa, prevaleciendo en caso de conflicto, el del último archivo importado.

La hoja de estilo se puede **importar mediante un string o una url**:

```
@import "navigation.css"; /* Using a string */

or

@import url("navigation.css"); /* Using a url */
```

A continuación, se presentan algunos **ejemplos**:

2) Importa la hoja de estilos “printstyle.css” SOLO si el tipo de media es ***print***:

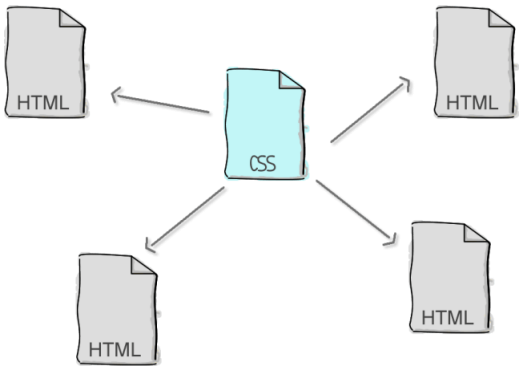
```
@import "printstyle.css" print;
```

3) Importa la hoja de estilos “mobstyle.css” SOLO si el tipo de media es ***screen*** y el ***viewport*** es de un máximo de 768 píxeles:

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

2.3.2 Hojas de estilo vinculadas o enlazadas.

Es el **modo más potente de utilizar CSS**. Se reúnen todas las reglas de estilo en un documento de texto independiente y se crean vínculos a ese documento desde todas las páginas de un sitio.



De este modo, **se pueden hacer cambios en el estilo de un modo homogéneo** en todo el sitio editando la información del estilo en un único documento.

Para hacerlo se emplea el elemento ***link*** en el documento (X)HTML. Este elemento se coloca en la **cabecera** (***head***) del documento.

Hay que tener en cuenta que **será muy importante el orden en el que se pongan, pues se aplicarán "de arriba hacia abajo"**, es decir, la última hoja de estilos enlazada modificaría los estilos que coincidan con los de las anteriores.

~/Desktop/Libro Diseño Interfaces WEB/cap4/cap4.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <title>
        Ejemplo de estilos CSS en un archivo externo
    </title>
    <link rel="stylesheet" type="text/css" href="guia.css" media="screen" />
</head>
<body>
    <p>Un párrafo de texto.</p>
</body>

</html>
```

```
<link rel="stylesheet" type="text/css" href="guia.css" media="screen" />
```

~/Desktop/Libro Diseño Interfaces WEB/cap4/guia.css

```
p{
color: red;
font-family:
Verdana;
}
```

La **función de cada uno de los atributos** usados para enlazar el documento HTML con la hoja de estilos en CSS es:

⇒ ***rel***. Indica el **tipo de relación** que existe entre el recurso enlazado, el archivo CSS y la página HTML. Para los archivos CSS, el valor es ***'stylesheet'***.

⇒ ***type***. Indica el **tipo de recurso enlazado**, en el caso de los archivos CSS su valor es "***text/css***".

⇒ ***href***. Recoge la **URL del archivo CSS**, esta puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.

⇒ ***media***. Indica el **tipo medio** en el que se van a aplicar los estilos del archivo CSS. Más adelante se explican en detalle los medios CSS y su funcionamiento.

En la actualidad, el uso de CSS en HTML está muy extendido, puesto que a través de un solo archivo CSS es posible incluir **una misma guía de estilo en todas las páginas HTML** de un mismo sitio web.

A continuación, se muestra un **ejemplo**:

⇒ El código **HTML**:

```
<html>
	<head>
		<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
		<title>Ejemplo de estilos CSS en el propio documento </title>
		<link rel="stylesheet" type="text/css" media="screen, handheld"
		href="hojaEst.css">
	</head>
	<body>
		<p>Un párrafo de texto.</p>
	</body>
</html>
```

⇒ El código **CSS** (***hojaEst.css***):

```
p {
	color: red; font-family: Verdana;
}
```

¿Cuáles son las **diferencias principales** entre utilizar `@import` o `<link>` para una hoja de estilos externa?

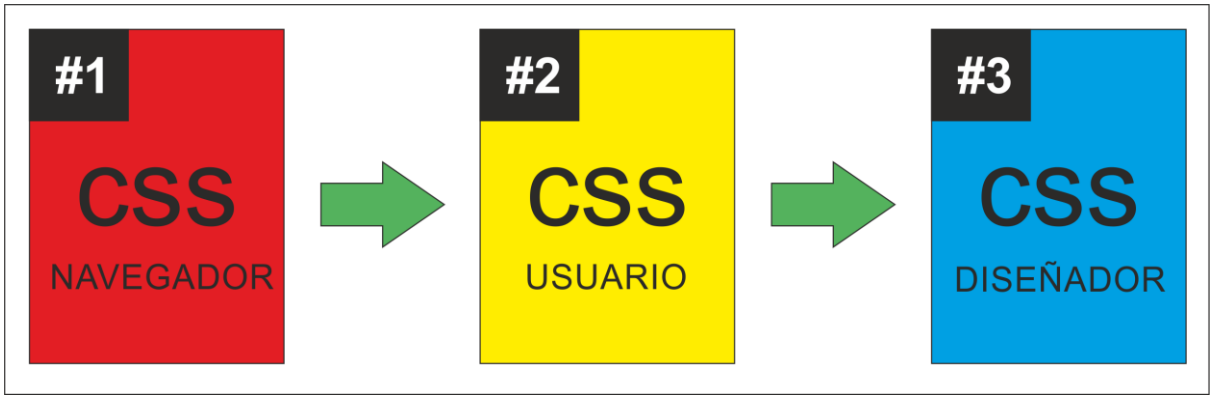
1) En primer lugar, `@import` se utiliza para incrustar la **hoja de estilo externa**, y por ello se pone dentro de la etiqueta html `<style>`. La etiqueta `<link>` se utiliza para enlazar la hoja de estilo externa.

2) Si usamos `@import` debemos **tener en cuenta** lo siguiente:
   - `@import` no descarga una hoja de estilos .css si no ha terminado con la anterior.
   - Si tenemos muchos `@import`, los archivos no siempre se descargan en orden.
   - `@import` se empezó utilizando porque tenía mayor compatibilidad con navegadores web antiguos.

3) Algunas de las **ventajas** de utilizar `<link>`, respecto de `@import`:
   - Nos aseguramos de que los recursos sean descargados en el **orden especificado**.
   - Las hojas de estilos se descargan casi en **paralelo** (no se espera a descargar el anterior para empezar con el siguiente) a través de todos los navegadores.

3. Orden de aplicación

Cuando escribimos código CSS, dicho código no será el único que se aplicará a la página web, de manera que se aplicarán antes los estilos configurados en el navegador web, y posteriormente las personalizaciones que el usuario haya querido hacer en el navegador.



Por tanto, **los navegadores web implementan el primer estilo**, ya que estos tienen un sistema de interpretación de etiquetas en la que se le asignan unos estilos por defecto. Dicha configuración la podemos ver en *Opciones>Contenido* para Firefox, o en *Configuración>Aspecto* en Chrome.

3.1 Normalize

Para evitar que se apliquen estilos indeseados por parte del navegador web y del usuario, se recomienda utilizar un archivo **CSS normalize** o reset CSS. Actualmente el más utilizado es el que podrás encontrar en la siguiente url: https://necolas.github.io/normalize.css/.

El archivo normalize o reset CSS conoce las características de los navegadores, corrigiendo errores e inconsistencias, y haciendo que las configuraciones de estilo de estos no apliquen o no afecten a nuestro diseño web. Para ello, el archivo normalize.css **deberá ir enlazado antes de nuestra hoja de estilos**:

```
<head>

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Titulo</title>
    <link rel="stylesheet" href="css/normalize.css">
    <link rel="stylesheet" href="css/estilos_con_responsive.css">

</head>
```

4. Prefijos

Existen atributos CSS que para poder implementarse en algunos de los navegadores web, necesitamos especificar exactamente que sea posible reproducirse en ese navegador, puesto que dicha propiedad CSS no es propia y puede ser problemática (presenta dificultades para generar el estilo que queremos).

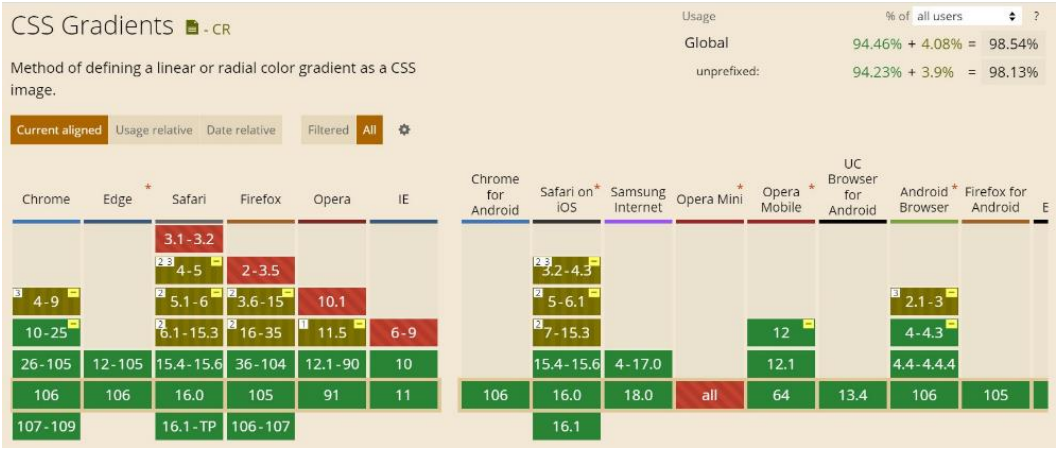
¿Y cómo sabemos que características CSS pueden ser problemáticas y en qué navegador web? Para ello podemos usar el buscador de la página https://caniuse.com/

Hay algunas propiedades CSS que están soportadas en los navegadores web, siempre y cuando utilicemos los siguientes prefijos:

- **-webkit-**, para Google Chrome, Safari y el navegador de Android.
- **-moz-**, para mozilla Firefox.
- **-o-**, para Opera.
- **-ms-** para Internet Explorer y Edge.
- **-Khtml-**, para Konqueror.

Un **ejemplo** de aplicación sería el siguiente, en el que queremos usar el atributo linear-gradient para generar un fondo con degradado lineal.

```
.linear-gradient {
  background-image: -moz-linear-gradient(top, #D7D 0%, #068 100%);
  background-image: -o-linear-gradient(top, #D7D 0%, #068 100%);
  background-image: -webkit-linear-gradient(top, #D7D 0%, #068 100%);
  background-image: linear-gradient(top, #D7D 0%, #068 100%);
}
```



5. Medios

Las hojas de estilo permiten especificar **cómo se representa un documento, en función del tipo de medio de visualización**.

Se denominan **medios** a los **canales a través de los cuales se muestra la información** del sitio web: auditivos, visuales, impresos, etc. Lo habitual es que se utilice una propiedad definida para cada tipo de canal. En la actualidad, **el medio más común** es el utilizado para definir el aspecto del sitio web en **pantalla**, `screen`, y el que define la página para ser impresa, `print`.

Para especificar el medio de destino se utilizan varios mecanismos, desde la hoja de estilo en CSS o desde el propio fichero HTML del sitio web.

Medio	Descripción
all	Todos los medios definidos.
print	Impresoras.
braille	Dispositivos táctiles que emplean el sistema braille.
embosed	Impresoras braille.
handheld	Dispositivos de mano: móviles, PDA, etc.
projection	Proyectores y dispositivos para presentaciones.
screen	Pantallas de ordenador.
speech	Sintetizadores para navegadores de voz utilizados por personas discapacitadas.
tv	Televisores y dispositivos con resolución baja.

5.1 @media

La regla `@media` especifica los **tipos de medios de destino**. A través de esta opción es posible que, en una misma hoja de estilo, aparezcan reglas para distintos medios. Si un mismo estilo se aplica a varios medios, basta con indicar sus nombres separados por comas.

- Sintaxis de `@media`

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

En el siguiente diagrama de código se muestra un **ejemplo** de uso de ***@media*** para indicar el tipo de medio donde se va a visualizar el sitio web. En este caso**, se observan dos tipos de medio, impresión y visualización por pantalla**. El tamaño de la letra se especifica para cada medio por separado, mientras que para el interlineado se va a utilizar el mismo en ambos casos, por lo tanto, se puede indicar de forma conjunta.

```
@media print {
	h1 {
		font-size: 15pt;
	}
}
@media screen {
	h1 {
		font-size: 15px;
	}
}
@media screen, print {
	h1 {
		line-height: 1.2;
	}
}
```

5.2 @import

A través de ***@import*** es posible **importar nuevas reglas desde otros ficheros CSS**, e indicar el medio o medios sobre los que debe aplicarse el nuevo estilo.

A diferencia de lo que ocurría con ***@media***, en primer lugar, aparece la URL que apunta al fichero CSS que contiene la guía de estilo, y, a continuación, se indica el medio sobre el que se va a aplicar. Es importante **tener en cuenta que estas reglas deben preceder a todas las demás reglas.**

Si los datos de estilo del fichero se van a aplicar a varios medios al mismo tiempo, se indicará el nombre de estos separados por comas.

⇒ Tal y como se indicó anteriormente, la sintaxis de ***@import*** es:

```
<head>
    <style>
        @import url|string lista-de-mediaqueries-o-nombreTipoMedio;
    </style>

</head>
```

⇒ **Ejemplo** de código:

```
@import "ficheroEstilo1.css" print;
```

```
@import "ficheroEstilo2.css" screen;
```

```
@import "ficheroEstilo3.css" print, screen;
```

```
@import "fichero Estilo4.css"; (Por defecto se aplica a todo)
```

En el ejemplo anterior, las dos primeras reglas aplicarán los estilos contenidos en los ficheros "*ficheroEstilo1.css*" y *"ficheroEstilo2.css"*, en medio impreso y por pantalla, respectivamente. La tercera regla aplicará el estilo de *"ficheroEstilo3.css"* en ambos medios. Finalmente, la última regla, al no indicar ningún medio, se aplicará en todos (all).

Es posible encontrar las reglas anteriores combinadas entre sí en un mismo fichero escrito en HTML. Por **ejemplo**, anidando las siguientes líneas:

⇒ Utilizando ***'link'***:

```
<link rel="stylesheet" type="text/css" media="screen" href="basico.css"/>
```

⇒ Utilizando '***@import'***:

```
@import url("estilos_seccion.css") screen;
```

⇒ Utilizando '***@media'***:

```
@media print {
      /* Estilos específicos para impresora */
}
```

5.3 Hojas de estilo auditivas, CSS Speech Module

Las hojas de estilo auditivas **proporcionan información para usuarios invidentes** y de navegadores de voz de manera parecida a la que se proporciona visualmente.

Las propiedades CSS definidas en el módulo del habla (CSS Speech Module) permiten a los autores controlar de forma declarativa la presentación de un documento en la versión auditiva.

El procesamiento auditivo de un documento **combina la síntesis de voz** (también conocido como «**TTS», el acrónimo de «Text to Speech**«) **y los iconos auditivos** («audio cues» en esta especificación).

Las propiedades CSS del habla proporcionan la **capacidad de controlar el tono del habla y velocidad, el volumen, voces TTS utilizadas**, etc.

Estas propiedades CSS **pueden ser utilizadas junto con las propiedades visuales** (medios mixtos), o **como una alternativa completa fonética a una presentación visual**.

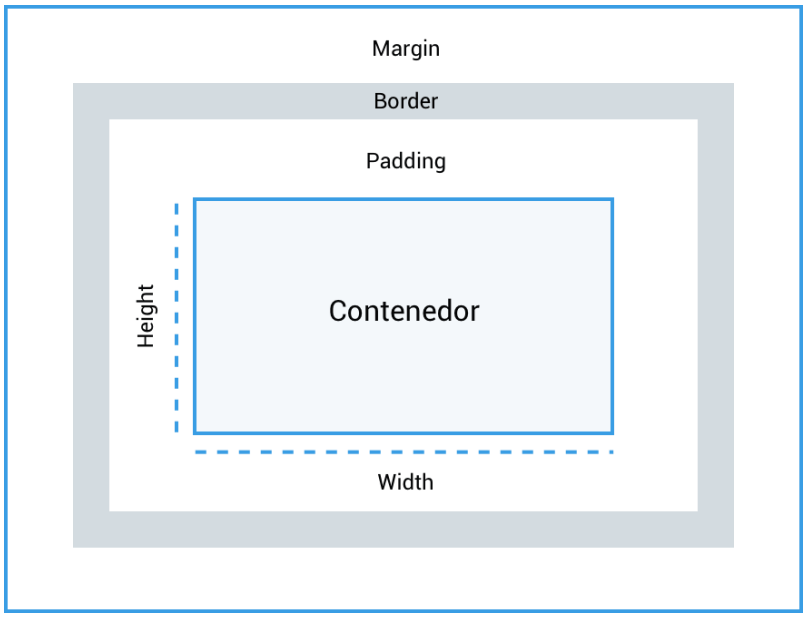
Se puede leer más sobre este tipo de hojas de estilo, en el siguiente artículo:

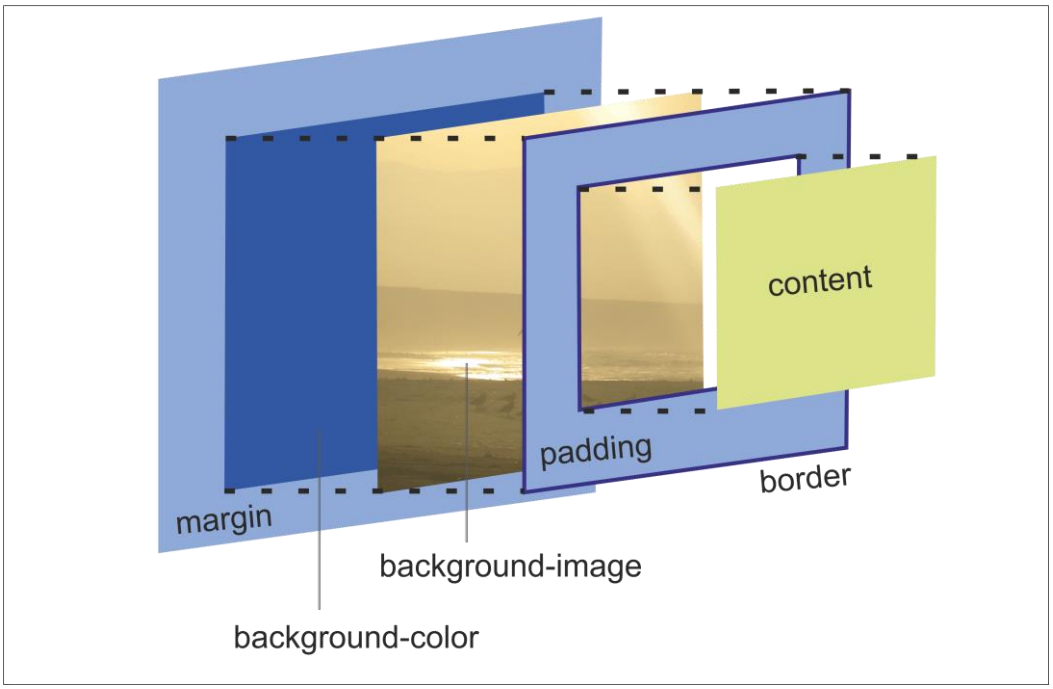
https://escss.blogspot.com/2012/06/css-speech-module-css-hablado.html

6. El modelo de cajas CSS

El **modelo de cajas** es una de las características más importantes de las hojas de estilos. Cuando se crea una página en HTML, **el uso de cada etiqueta simula lo que podría considerarse una caja**, dotando a cada una de ella de un identificador, indicado por el nombre de la etiqueta.

Este sistema de cajas **permite**, desde el fichero CSS, **seleccionar cada caja y dotarla del estilo deseado.**





El uso de las guías de estilo con CSS permite modificar todas las características de cada una de las cajas creadas. A continuación, se analizan cada una de estas partes a través de un ejemplo que irá incorporando nuevas propiedades.

6.1 CONTENT.

Las **propiedades** principales con las que se trabaja en esta ***caja de contenido*** o ***content***, son relativas a sus dimensiones. Dichas propiedades son ***width*** y ***height***, que hacen referencia al **ancho y el alto**, respectivamente, del área donde se muestra el contenido de la caja.

Otra de las propiedades más importantes del área de contenido es el **fondo** de la caja. Se diferencia entre:

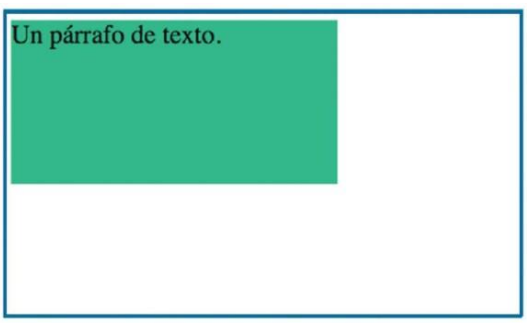
- ***background image*** (imagen de fondo), que corresponde a la imagen que se muestra por detrás del contenido y el espacio de relleno; y

- ***background color*** (color de fondo), que es el color que se muestra por detrás del contenido y el espacio de relleno.

Por **ejemplo**, si se introduce este código:

```
article {
      background: #5DEAAB;
      height: 100px;
      width: 200px;
}
```

En pantalla, se verá:



Existen otras propiedades que permiten el manejo refinado del tamaño de la caja de contenido para limitar los tamaños. Para ello se usan las **propiedades** ***min-width***, ***min-height***, y ***max-height***.

6.2 PADDING.

Esta segunda área es la de relleno, llamada **padding**. Es el **espacio que existe entre la caja de contenido y el borde**. Por tanto, podemos decir **que el padding es un borde interno**; no es un borde con respecto a los elementos que hay alrededor.

Las **características** de esta caja, **que pueden modificarse son** su **altura** y su **anchura**. Se distinguen, por tanto, **cuatro zonas posibles de acción**: zona **inferior**, zona **superior**, zona **izquierda** y zona **derecha**. El nombre de las propiedades que definen estas zonas son ***padding-top***, ***padding-right***, ***padding-left***, y ***padding-bottom***.

```
article {
        padding-top: 10px;
        padding-right: 30px;
        padding-bottom: 10px;
        padding-left: 20px;
}
```

Si solo escribimos ***padding***, **se aplicará el mismo formato a todos los extremos**. En el siguiente **ejemplo**, los límites inferior, superior izquierdo y derecho tendrían un valor de 10 px.

```
article {
        padding: 10px;
}
```

Existen **más tipos de empleo de esta regla** que simplifican su uso, que se basan en el empleo de la palabra ***padding*** seguida de varias cifras separadas sin comas.

- Si aparecen **dos valores**, el primero es asignado al límite superior y al inferior, mientras que la segunda cifra será la que se aplique al relleno horizontal (lados).

```
article {
      padding: 10px 5px;
}
```

- top and bottom padding: 10px
- right and left padding: 5px

- En el caso de tener **tres valores**, el primero y el último serán los aplicados a los límites superior e inferior, respectivamente, y la segunda cifra indica el valor de los bordes izquierdo y derecho.

```
article {
    padding: 10px 5px 15px;
}
```

- top padding: 10px
- right and left padding: 5px
- bottom padding: 15px

- En el caso de tener **cuatro valores**:

```
article {
padding: 25px 50px 75px 100px;
}
```

- top padding: 25px
- right padding: 50px
- bottom padding: 75px
- left padding:100px

Los padding NO se utilizan para separar elementos en CSS. Esta tarea se realiza con los **margin.** Los padding simplemente sirven para **dar más grosor a un elemento.** Por **ejemplo**, si le ponemos un buen padding a un enlace, haremos que la parte clicable sea más grande.

6.3 BORDER.

La siguiente área que puede verse en el diagrama de cajas es la correspondiente al borde, llamado ***border***. Esta zona es la que **encierra el contenido y el relleno**, *content y padding*.

Es posible modificar su grosor, estilo y color. Además**, al igual que ocurre con el padding, es posible dar formato a los cuatro extremos de la caja a la vez** utilizando solo la palabra border o hacerlo de forma individual.

```
article {
    border: 1px solid #000000;
}
```

En aquellas ocasiones en las que se desee dar un formato diferente a cada uno de los extremos, se utilizarán algunas de las propiedades individuales destinadas a tal fin.

Propiedad	Descripción	Valores	Ejemplo
border	*Modifica a la vez el grosor, el estilo, y el color de todos los lados del borde.*	border-width | border-style | border-color	h2 { border: 5px solid red; }
border-top ***border-right*** ***border-bottom*** ***border-left***	*Modifican a la vez el grosor, el estilo y el color de cada lado del borde.*	border-top-width | border-top-style | border-top-color	h1 { border-top: 5px solid red; }
border-width	*Modifica el grosor de los cuatro extremos a la vez.*	thin | medium | thick | inherit | un valor concreto en píxeles.	div { border-style: solid; border-width: thin medium thick 12px; }
border-style	*Modifica el estilo de los cuatro extremos a la vez.*	none | hidden | dotted | dashed | solid | double | groove | ridge |inset | outset {1,4} | inherit.	div {border-style: solid dashed dotted double; } div {border-style: solid dashed dotted double; }
border-color	*Modifica el color de los cuatro extremos a la vez. Si no especificamos el color el elemento coge el del "primer plano", es decir que si, por ejemplo, tenemos una caja en cuyo interior hay texto,*	color | transparent {1,4}	div { border-style: solid; border-width: 4px; border-color: #333 #red rgb(0,0,255) #0044AC; }

	el color del borde será el color del texto. Existe el color `transparent` *pero no está soportado por todos los navegadores.*		
border-top|right|bottom|left-width	*Modifica el grosor del borde superior, derecho, inferior o izquierdo.*	`thin` | `medium` | `thick` | un valor concreto en `píxeles`.	`div { border-top-width: thick; }`
border-top|right|bottom|left-style	*Modifica el estilo del borde superior, derecho, inferior o izquierdo.*	`none` | `hidden` | `dotted` | `dashed` | `solid` | `double` | `groove` | `ridge` |`inset` | `outset`	`div { border-top-style: solid; }`
border-top|right|bottom|left-color	*Modifica el color del borde superior, derecho, inferior o izquierdo.*	`color` | `transparent`	`div { border-top-color: red; }`
border-radius	*Modifica la curvatura del borde.*	`longitud` | `porcentaje {1,4}`	`div { border-radius: 30px; }`

6.4 MARGIN.

Finalmente, el margen que **envuelve al resto de elementos CSS** es el denominado ***margin***, y sostiene a otras cajas del diseño.

Las propiedades individuales son ***margin-top***, ***margin-right***, ***margin-bottom*** y ***margin-left***. **El funcionamiento es igual que el de la propiedad *padding*.**

```
article {
      margin-top: 10px;
      margin-right: 10px;
      margin-bottom: 10px;
      margin-left: 10px;
}
```

Los márgenes tienen un comportamiento peculiar llamado **margin collapsing**. **Cuando dos cajas se tocan, la distancia entre ellas es el valor del margen más grande, y no la suma de ambos.**

6.5 box-sizing: border-box

El valor ***border-box*** en CSS para la propiedad ***box-sizing*** nos permite trabajar más cómodamente con el modelo de cajas de CSS.

La propiedad de CSS ***box-sizing*** **indica cómo se deben calcular las medidas de un elemento**. Su valor por defecto es ***content-box***. Esto, que parece trivial, no lo es tanto ya que por defecto **CSS considera que el ancho y alto de la caja es de las propiedades width y height**. ¿Qué significa esto? Pues que **si le añades un padding o un border el tamaño de renderizado de la caja será**:

width + padding + border.

En el siguiente **ejemplo** tendríamos una **caja de 290px de ancho** ya que: *250px de width + (10px * 2) de padding + (10px * 2) de border.*

```
div {
  width: 250px;
  border: 10px;
  padding: 10px;
}
```

Esto es bastante problemático ya que hace que sea muy difícil de calcular de forma predecible el ancho o alto de nuestros elementos si tienen padding o border. Para arreglar esto tenemos el valor ***border-box***, el cual tiene soporte total en los navegadores de hoy en día. Para saber si border-box tiene **soporte** con nuestro navegador, podemos utilizar la página web https://caniuse.com/

El valor ***border-box*** en el ***box-sizing*** hace que el padding y el border pasen a formar parte del cálculo del ancho de la caja y no lo suman posteriormente.

En el siguiente **ejemplo** tendríamos una **caja de 250px de ancho** ya que el padding y el border ya forman parte del cálculo del width del elemento:

```
div {
  box-sizing: border-box;
  width: 250px;
  border: 10px;
  padding: 10px;
}
```

Muchos resets CSS o normalizers ya incluyen el CSS necesario para que nuestra página use este tipo de cálculo del modelo de la caja de CSS pero si, por lo que fuese, necesitamos añadirlo manualmente en nuestro proyecto, **podemos usarlo de la siguiente forma**:

```
* {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}
```

6.6 La propiedad 'resize'

La propiedad ***resize*** define si (y cómo) un elemento es redimensionable por el usuario.

Esta propiedad no se aplica a elementos 'inline' o a elementos de bloque donde *overflow="visible"*. Por lo tanto, asegúrate de que ***overflow*** está configurado como ***"scroll"***, ***"auto"*** o ***"hidden"***.

Su **sintaxis** es la siguiente:

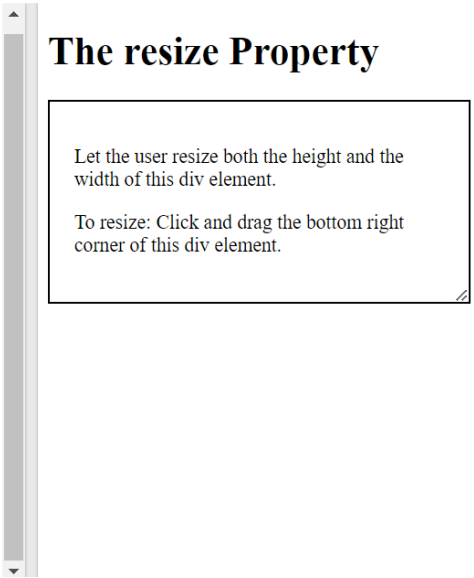
```
etiqueta html {
      resize: none | both | horizontal | vertical | initial | inherit;
      }
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 2px solid;
  padding: 20px;
  width: 300px;
  resize: both;
  overflow: auto;
}
</style>
</head>
<body>

<h1>The resize Property</h1>

<div>
  <p>Let the user resize both the height and the width
of this div element.</p>
  <p>To resize: Click and drag the bottom right corner
of this div element.</p>
</div>

</body>
</html>
```



7. Herencia

La **herencia en CSS** es el mecanismo mediante el cual determinadas propiedades de un elemento "padre" se transmiten a sus "hijos". Cuando se establece el valor de una propiedad CSS en un elemento, sus elementos **descendientes heredan de forma automática el valor de esa propiedad**.

⇒ Se dice que **un elemento es "hijo" de otro si está contenido directamente en él y este último pasa a ser su "padre"**. Por ejemplo: el elemento ***<p>*** es hijo del elemento ***<body>***y el elemento ***<body>*** es padre del elemento ***<p>***.

⇒ **Los elementos que tienen el mismo padre son "hermanos"**. Por ejemplo: un elemento ***<p>*** puede ser hermano de otro elemento ***<p>*** si ambos son hijos directos del elemento ***<body>***.

Controlar la relación padre-hijo es fundamental para el funcionamiento de CSS. Con una buena planificación, **la herencia puede emplearse para hacer más eficiente la especificación de los estilos**.

No todas las propiedades CSS son heredables; los márgenes no se heredan porque es poco probable que un elemento "hijo" necesite los mismos márgenes que su "padre".

```
<html>
	<head>
		<title>Ejemplo de herencia de estilos</title>

		<style type="text/css">
			body { color: blue; }
		</style>
	</head>

	<body>
		<h1>Titular de la página</h1>
		<p>Un párrafo de texto no muy largo.</p>
	</body>
</html>
```

En el ejemplo, el selector ***body*** solo establece el color de la letra para el elemento ***<body>***. Esta propiedad se hereda de forma automática, ¿qué significa esto? Que los elementos descendientes muestran ese mismo color. Ahora bien, **la herencia de estilos se aplica automáticamente, pero su efecto puede ser anulado** estableciendo de forma explícita otro valor para la propiedad que se hereda.

Existen **otras propiedades cuyo valor no se hereda**, como es el caso de los **bordes** (***border***); es decir, si se define un estilo relativo al borde para el cuerpo, este no será aplicado al resto de elementos HTML.

Algunas **propiedades que sí son heredadas** por defecto:

color	text-align
font	visibility
letter-spacing	white-space
direction	border-collapse
word-spacing	quotes
line-height	list-style-image

Ahora bien, como ya se ha dicho, existen ciertas propiedades que no son heredables de "padres" a "hijos", pero **existe un mecanismo mediante el cual es posible indicar que un elemento herede el valor de una propiedad**. Para ello se utilizan los **valores**: ***inherit***, ***initial*** y ***unset***, que **modifican el valor de la propiedad respecto a la herencia**.

a) ***inherit***. Aplica el valor de la propiedad del elemento "padre".
b) ***initial***. Aplica el valor que presenta inicialmente la propiedad.
c) ***unset***. Hereda el valor de la propiedad del elemento "padre"; si este no existe, aplica el valor inicial.

```
body {
      border-width: 10px;
      border-color: blue;
      border-style: solid;
      color: red;
      }
h1 {
      border: inherit;
      }
```

En el ejemplo anterior, el elemento ***h1*** hereda todas las propiedades establecidas en el ***body***. El color se hace forma automática, puesto que es una de las propiedades que se heredan, y al no establecerse otro valor para el color en ***h1*** quedará de color rojo. Ahora bien, en este ejemplo las propiedades de borde también serán heredadas puesto que así es indicado por el valor ***inherit***.

8. Reglas de estilo en conflicto

Cuando un navegador encuentra un elemento para el cual hay varias declaraciones de estilo, las ordena de acuerdo al origen de la hoja de estilo, la especificidad de los selectores y el orden de la regla para poder determinar cuál aplicar.

Existe una **jerarquía de pesos** que se aplican a las hojas de estilo creadas por la persona que ha diseñado la página web. Es importante entender esta jerarquía y tener en cuenta que las reglas de estilo que están al final de la lista ignorarán a las primeras. La siguiente lista, está ordenada **de menor a mayor peso**:

1) Estilos del **navegador web** por defecto.
2) Estilos configurados por el **propio usuario en su navegador web**.
3) **Hojas de estilo externas vinculadas** (empleando el elemento ***link*** en la cabecera del documento), **en el orden en que hayan sido vinculadas**.
4) Hojas de **estilo externas importadas** (empleando el elemento ***@import*** dentro del elemento ***style*** en la cabecera del documento), en el orden en que hayan sido vinculadas.
5) **Estilos en línea** (empleando el atributo ***style*** en la etiqueta del elemento).
6) **Declaraciones de estilo marcadas como** ***!important***, la cual es usada para añadir más importancia de la normal a una propiedad o valor, anulando todas las reglas de estilo anteriores para dicha propiedad.

```
p {
  background-color: red !important;
}
```

Por otro lado, cuando una hoja de estilo contiene **varias reglas en conflicto de igual peso**, sólo se tendrá en cuenta la que está en **último lugar**. En el siguiente ejemplo, todas las cabeceras de primer nivel del documento serían rojas porque se impone siempre la última regla:

```
h1 {color: green;} h1 {color: blue;} h1 {color: red;}
```

9. Consejos para un desarrollo ordenado.

Escribir un desarrollo ordenado del código es una buena práctica. Cuando se escribe código, es muy complicado de mantener, o de poder seguir extendiéndolo si se encuentra desordenado.

Mantén un código que sea fácil de entender, no solamente para ti, sino también para otras personas. Para ello, sigue las siguientes **recomendaciones**:

1) Procura que el código HTML/CSS se encuentre bien **identado**. Se llama **identación** de código al hecho de usar sangrado (mover ligeramente a la derecha) en las líneas de código para facilitar la lectura, e indicar visualmente si nos encontramos en el interior de una función, bucle, condicional, etc. Por tanto, la identación nos permite ver dónde hay un elemento padre, un elemento hijo, donde un elemento es más interno, etc.

2) Intenta que los **identificadores** (*id*) y las **clases** (*class*) sean **descriptivos y coherentes**.

3) Realiza **anotaciones globales** al inicio del código CSS en forma de comentario para toda la página.

4) Define los estilos en **CSS según el orden de los elementos HTML** que hayas escrito en tu hoja html. Aquellas partes más globales de los estilos, se pueden poner al inicio de la hoja CSS.

5) **Ordena** las propiedades CSS por **orden alfabético** ya que, si en un futuro necesitamos modificar alguna, nos ahorraremos tiempo de búsqueda.

6) Localiza las **media queries** para el Responsive Design la **final del archivo CSS**.