2.1. Introducción a hojas de estilo en cascada (CSS, Cascading Style Sheet)

CSS (**Cascading Style Sheets**, Hojas de Estilo en Cascada) es un **lenguaje de estilo utilizado para definir la presentación y el diseño de documentos HTML o de cualquier lenguaje de marcado basado en XML**. Con CSS, se puede controlar la apariencia de los elementos de una página web, como el color de fondo, la tipografía, el espaciado, los bordes y la disposición de los elementos en pantalla. Veamos detalladamente qué es CSS y cuál es su sintaxis.

El lenguaje **CSS está creado por el World Wide Web Consortium (W3C)**, la comunidad internacional que desarrolla estándares que aseguran el crecimiento futuro de la web y vela por conseguir webs disponibles para todo el mundo.

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**. A lo largo de todo el capítulo, se detallan las propiedades CSS más utilizadas en relación a los colores, el texto, los fondos, las listas, las tablas y el modelo de cajas.

2.1.1 Soporte CSS en los navegadores más utilizados

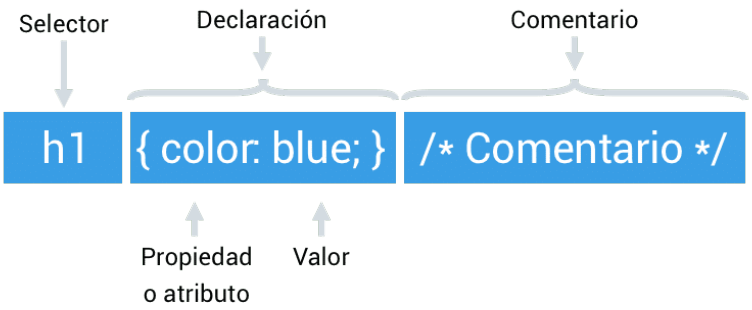
Cada navegador ofrece un soporte CSS distinto. Por este motivo, es necesario comprobar que nuestras webs se visualizan correctamente en los diferentes navegadores. A continuación, se presentan algunas herramientas útiles para verificar la compatibilidad y resolver problemas de CSS:

1. **Can I use** (caniuse.com): Esta herramienta en línea permite verificar el soporte de diferentes características de CSS en varios navegadores y versiones.
2. **Browserstack (**browserstack.com): Es una plataforma que permite probar sitios web en diferentes navegadores y dispositivos para verificar la compatibilidad de CSS.
3. **Developer Tools en los navegadores**: Chrome, Firefox, Safari y Edge ofrecen herramientas integradas de desarrollo que permiten inspeccionar y modificar el código CSS para solucionar problemas y ajustar el diseño.
4. **CSS Validator** (jigsaw.w3.org/css-validator/): Es una herramienta de la W3C que verifica la validez del código CSS y muestra errores y advertencias para corregir problemas.

El uso de estas herramientas y el conocimiento del soporte de CSS en los navegadores más utilizados son fundamentales para garantizar que un sitio web tenga un diseño coherente y atractivo en diferentes entornos y dispositivos.

2.1.2. 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.
- **Declaración**: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades 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 */. Pueden contener varias líneas.

2.1.3. Formas de aplicar estilos CSS

Hay tres formas de aplicar estilos CSS en un documento HTML: en línea, incrustado en la cabecera y mediante hojas de estilo externas.

1. CSS en línea

Los estilos en línea son declaraciones CSS que se integran en las etiquetas HTML mediante el atributo style. Este método tan solo afecta al elemento en el que se integra el código. El CSS en línea es **complicado de entender y mantener ya que mezcla los estilos CSS con el código HTML**.

Ejemplo:

```
<html>
   <head>
      <title>CSS integrado en las etiquetas</title>
   </head>
   <body>
      <p style="color:green;">Párrafo de color verde.</p>
   </body>
</html>
```

2. CSS incrustado en la cabecera o CSS interno

Otra manera muy simple de añadir estilo con CSS es utilizando la etiqueta <style> en la cabecera <head> del fichero HTML del sitio. 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**. Su uso puede llegar a ser necesario en el caso de utilizar un gestor de contenido que no permita modificar el archivo CSS directamente.

Ejemplo:

```
<html>
  <head>
    <title>CSS incrustado en la cabecera</title>
    <style> p { color: green; } </style>
  </head>
  <body>
    <p>Párrafo de color verde.</p>
  </body>
</html>
```

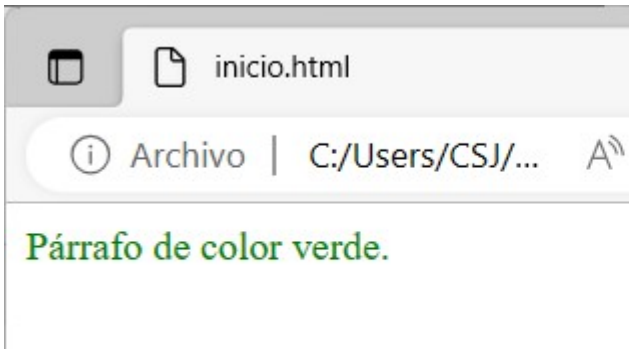
2.3. CSS en hojas de estilo externas

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

Ejemplo:

inicio.html

```
<html>
  <head>
    <link rel="stylesheet" href="styles.css">
  </head>
  <body>
    <p>Párrafo de color verde.</p>
  </body>
</html>
```



styles.css

```
p {color:green;}
```

2.1.4 Hojas de estilo múltiples.

Si se han definido algunas propiedades para el mismo selector (elemento) en diferentes hojas de estilo, se utilizará el valor de la última hoja de estilo de lectura.

EJEMPLO: Supongamos que una hoja de estilo externa tiene el siguiente estilo para el elemento <h1>:

```
h1{
 color:navy;
}
```

luego, imaginemos que una hoja de estilo interna también tiene el siguiente estilo para el elemento <h1>:

```
h1{
 color:orange;
}
```

Si el estilo interno se define después del enlace a la hoja de estilo externa, los elementos <h1> serán

"naranja":

EJEMPLO

```
<head>
<link rel="stylesheet" type="text/css" href="example.css">
<style>
h1{
 color: orange;
}
</style>
</head>
```

Sin embargo, **si el estilo interno se define antes del enlace a la hoja de estilo externa**, los elementos <h1> serán "navy":

EJEMPLO

```
<head>
<style>
h1{
 color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="example.css">
</head>
```

ORDEN EN CASCADA
¿Qué estilo se usará cuando hay más de un estilo especificado para un elemento HTML?

Todos los estilos en una página se "colocarán en cascada" en una nueva hoja de estilo "virtual" según las siguientes reglas, donde el número uno tiene la mayor prioridad:

1. Estilo en línea (dentro de un elemento HTML)
2. Hojas de estilo externas e internas (en la sección de cabeza)
3. Navegador predeterminado

Por lo tanto, **un estilo en línea tiene la prioridad más alta,** y reemplazará los estilos externos e internos y los valores predeterminados del navegador.

ACTIVIDAD 2.1

2.1.5 Hojas de estilo externas.

Las hojas de estilo son documentos de texto con, por lo menos, una regla. Estos archivos no contienen etiquetas HTML. Al igual que en los documentos HTML, en las hojas de estilo se pueden incluir comentarios, pero, en este caso, se escriben del siguiente modo: **/* Este es un comentario */**

Cuando el navegador carga la página HTML, antes de mostrar sus contenidos también descarga los archivos CSS externos enlazados mediante la **etiqueta <link>** y aplica los estilos a los contenidos de la página. Normalmente, la etiqueta <link> incluye cuatro atributos cuando enlaza un archivo CSS:

- **rel**: indica el tipo de relación que existe entre el recurso enlazado (en este caso, el archivo CSS) y la página HTML. Para los archivos **CSS**, siempre se utiliza el valor **stylesheet**
- **type**: indica el tipo de recurso enlazado. Sus valores están estandarizados y para los archivos CSS su

valor siempre es **text/css**

- **href**: indica la URL del archivo CSS que contiene los estilos. La URL indicada puede ser relativa o absoluta y puede apuntar a un recurso interno o externo al sitio web.
- **media**: indica el medio en el que se van a aplicar los estilos del archivo CSS. Una de las características más importantes de las hojas de estilos CSS es que **permiten definir diferentes estilos para diferentes medios o dispositivos:** pantallas, impresoras, móviles, proyectores, etc.

Un ejemplo: `<link rel="stylesheet" type="text/css" href="/css/estilos.css" media="screen" />`

Además, **CSS define algunas propiedades específicamente para determinados medios**, como por ejemplo la paginación y los saltos de página para los medios impresos o el volumen y tipo de voz para los medios de audio. Para ello se emplea el **atributo media del elemento link.** La siguiente tabla muestra el nombre que CSS utiliza para identificar cada medio y su descripción:

Medio	Descripción
all	Todos los medios definidos
braille	Dispositivos táctiles que emplean el sistema braille
embosed	Impresoras braille
handheld	Dispositivos de mano: móviles, PDA, etc.
print	Impresoras y navegadores en el modo *"Vista Previa para Imprimir"*
projection	Proyectores y dispositivos para presentaciones
screen	Pantallas de ordenador
speech	Sintetizadores para navegadores de voz utilizados por personas discapacitadas
tty	Dispositivos textuales limitados como teletipos y terminales de texto
tv	Televisores y dispositivos con resolución baja

Los medios **más utilizados** actualmente son **screen** (para definir el aspecto de la página en pantalla) y **print** (para definir el aspecto de la página cuando se imprime), **seguidos de handheld** (que define el aspecto de la página cuando se visualiza mediante un dispositivo móvil).

La gran ventaja de CSS es que **permite modificar los estilos de una página en función del medio en el que se visualiza**. Existen **cuatro formas diferentes** de indicar el medio en el que se deben aplicar los estilos CSS.

1- MEDIOS DEFINIDOS CON LAS REGLAS DE TIPO @media

Las reglas @media son un tipo especial de regla CSS que **permiten indicar de forma directa el medio o medios en los que se aplicarán los estilos incluidos en la regla.** Para especificar el medio en el que se aplican los estilos, se incluye su nombre después de @media. Si los estilos se aplican a varios medios, se incluyen los nombres de todos los medios separados por comas.

A continuación, se muestra un ejemplo sencillo:

```
@media print {
  body { font-size: 10pt }
}
@media screen {
  body { font-size: 13px }
}
@media screen, print {
  body { line-height: 1.2 }
}
```

El ejemplo anterior establece que el tamaño de letra de la página cuando se visualiza en una pantalla debe ser 13 píxel. Sin embargo, cuando se imprimen los contenidos de la página, su tamaño de letra debe ser de 10 puntos. Por último, tanto cuando la página se visualiza en una pantalla como cuando se imprimen sus contenidos, el interlineado del texto debe ser de 1.2 veces el tamaño de letra del texto.

2. MEDIOS DEFINIDOS CON LAS REGLAS DE TIPO @import

Cuando se utilizan reglas de tipo @import para enlazar archivos CSS externos, **se puede especificar el medio en el que se aplican los estilos indicando el nombre del medio después de la URL** del archivo CSS:

```
@import url("estilos_basicos.css") screen;
@import url("estilos_impresora.css") print;
```

Las reglas del ejemplo anterior establecen que cuando la página se visualiza por pantalla, se cargan los estilos definidos en el primer archivo CSS. Por otra parte, cuando la página se imprime, se tienen en cuenta los estilos que define el segundo archivo CSS.

Si los estilos del archivo CSS externo deben aplicarse en varios medios, se indican **los nombres de todos los medios separados por comas**. **Si no se indica** el medio en una regla de tipo @import, el navegador sobreentiende que **el medio es all,** es decir, que los estilos se aplican en todos los medios.

3. MEDIOS DEFINIDOS CON LA ETIQUETA <link>

Si se utiliza la etiqueta <link> para enlazar los archivos CSS externos, se puede utilizar el atributo media para indicar el medio o medios en los que se aplican los estilos de cada archivo:

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

En este ejemplo, el primer archivo CSS se tiene en cuenta cuando la página se visualiza en la pantalla (media="screen"). Los estilos indicados en el segundo archivo CSS, se aplican al imprimir la página (media="print") o al visualizarla en un dispositivo móvil (media="handheld"), como por ejemplo en un iPhone.

Si la etiqueta <link> no indica el medio CSS, se sobreentiende que los estilos se deben aplicar a todos los medios, por lo que es equivalente a indicar media="all".

4. MEDIOS DEFINIDOS MEZCLANDO VARIOS MÉTODOS

CSS también permite **mezclar los tres métodos anteriores** para indicar los medios en los que se aplica cada archivo CSS externo:

```
<link rel="stylesheet" type="text/css"  media="screen" href="basico.css" />
@import url("estilos_seccion.css") screen;
@media print {
  /* Estilos específicos para impresora */
}
```

Los estilos CSS que se aplican cuando se visualiza la página en una pantalla se obtienen mediante el recurso enlazado con la etiqueta <link> y mediante el archivo CSS externo incluido con la regla de tipo @import. Además, los estilos aplicados cuando se imprime la página se indican directamente en la página HTML mediante la regla de tipo @media.

2.1.6 HERRAMIENTAS PARA CREAR CSS. Test de verificación.

Antes de que existieran herramientas avanzadas como ocurre en la actualidad, los desarrolladores debían escribir todo el código en un editor de texto, lo que los llevaba a cometer errores y tener dificultad para corregirlos. Estos casos podían ser:

- El selector podría estar mal escrito.
- Las propiedades están mal escritas o tienen valores no permitidos.
- Otros selectores tienen más prioridad y sobrescriben una propiedad o valor.

Muchos editores actuales proporcionan **herramientas que aportan facilidades para escribir código** CSS, por ejemplo Brackets, Visual Studio Code, etc.

Utilizar los recursos más apropiados es imprescindible para desarrollar con el mayor éxito cualquier página web. A continuación se describen **varias herramientas útiles** que conviene tener en cuenta a la hora de detectar errores en los desarrollos, mejorar la productividad y conseguir los mejores diseños.

1. CSS Validation Service, W3C

Tal y como se comentó en el primer punto de la presente unidad, el W3C es el encargado de estandarizar los estilos CSS y en su plataforma nos ofrece varias herramientas e información muy valiosa para la elaboración de nuestros desarrollos. Entre las herramientas destacadas se encuentra el validador CSS que nos permite encontrar errores en nuestros ficheros.

2. Browserling

Browserling una página que nos permite probar cualquier web en los diferentes navegadores: Internet Explorer, Firefox, Chrome, Opera y Safari, en distintas versiones y resoluciones de pantalla.

3. Caniuse

Puedes ver los navegadores que soportan una determinada propiedad CSS o un elemento HTML5

en la página web caniuse.com

4. Extensión Autoprefixer para Visual Studio Code

Para ahorrar tiempo y facilitarnos la tarea de incluir los prefijos de las propiedades CSS que todavía no son estables podemos hacer uso de la extensión “Autoprefixer” en Visual Studio Code.

5. Repositorio de fuentes de Google Fonts

En la página de Google Fonts tenemos a nuestra disposición un amplio repositorio de tipografías que podemos utilizar en nuestros desarrollos.

6. Repositorio de iconos de Font Awesome

Al igual que con las fuentes, podemos utilizar repositorios de iconos para representar cualquier parte de nuestra interfaz. Un ejemplo de este tipo de repositorios es Font Awesome.

7. Convertidores de fuentes online

Nombre	**URL**
Font Squirrel converter	fontsquirrel.com/tools/webfont-generator
Online Font Converter	onlinefontconverter.com
Files conversion	files-conversion.com/font-converter.php
Font converter	fontconverter.org

8. Fuentes web online

Nombre	**URL**
Font Squirrel	fontsquirrel.com
Da Font	dafont.com
Google Fonts	google.com/fonts

2.1.7 BUENAS PRÁCTICAS CSS.

Algunas recomendaciones a tener en cuenta al crear las reglas css:

- /* hay que poner **comentarios** */
- Los selectores se nombran en **minúsculas**, nunca empezando por caracteres especiales o numéricos. En las últimas versiones **se puede utilizar en el nombre “-”** no “_” ya que no todos los navegadores lo soportan.
- El nombre de los selectores debe ser **específico y claro**, para que tenga una mayor capacidad expresiva.
- El nombre de las clases e identificadores no debe describir una característica visual, como

color, tamaño o posición.

- Los nombres deben seguir más una **visión semántica** que estructural. Para facilitar los cambios.
- **Separa las palabras mediante guiones o mayúsculas.**
- **No** hacer **uso excesivo de clases**.
- **Agrupar las reglas según su selector siempre que sea posible**. Agruparlas unas debajo de otras:

```
[…]
table {border:double}
table.miembros {border:solid}
table.empleados{border:grrobe}
[…]
```

- **Al principio de un CSS es aconsejable definir los selectores de etiquetas**. Además de usar comentarios para dejar claro cuál es la parte que definen las clases y otros elementos.

```
/* Etiquetas html */
html {font-family:Arial, verdana, sans serif; font-size:13px;}
h1,h2,h3,h4,h4,h6,form,input,text-area{
border:0; padding:0; margin:0;
font-family:arial;}
/* FIN de Etiquetas HTML
```

- **Estructurar visualmente los atributos**. Si un elemento solo tiene 3 atributos se pueden poner en la misma línea. Pero si hay más se ponen en líneas diferentes sangrados con tabuladores.

Seguir unas pautas a la hora de crear estilos CSS hace que el código sea más claro y legible tanto para nosotros como para otros desarrolladores que deban trabajar en el proyecto.
Aunque no hay definido ningún estándar al respecto, es recomendable seguir las siguientes recomendaciones para conseguir un desarrollo lo más entendible posible.

1 Organizar la estructura de arriba hacia abajo

Con el fin de encontrar rápidamente cualquier parte de código, es recomendable organizar los estilos de arriba hacia abajo.

```
/****** generic classes*******/
styles goes here…
/****** header *******/
styles goes here…
/****** nav menu *******/
styles goes here…
/****** main content *******/
styles goes here…
/****** footer *******/
```

2. Nombrar correctamente los selectores

Para conseguir más claridad en el código y soporte en todos los navegadores conviene no comenzar el nombre de los selectores con mayúsculas, números ni caracteres especiales.

Evitar: ~~#1div~~, ~~.=div~~, ~~DivContent~~

Mejor utilizar: #div1, .div, divContent

3. Separar las palabras mediante guiones o mediante mayúsculas

Es recomendable escoger una única manera de escribir el nombre de los selectores. Además, es mejor no usar guiones bajos "_" u otros caracteres especiales ya que algunos navegadores no los soportan. Conviene seguir alguna de las siguientes opciones:

```
/* Opción 1: Palabras separadas por mayúsculas */
navMenu { padding: 2em; border: 2px solid green; }

/* Opción 2: Palabras separadas por guiones */
nav-menu { padding: 2em; border: 2px solid green; }
```

4. Legibilidad

Adopta una única forma de escribir tu código para que sea más fácil de mantener y de encontrar cualquier elemento.

```
/* Opción 1: Estilos en una línea */
.nav-menu { padding: 2em; border: 2px solid green; }

/* Opción 2: Cada estilo en una línea */
.nav-menu {
    padding: 2em;
    border: 2px solid green;
}
```

5. Combinar elementos

Cuando varios elementos disponen de las mismas propiedades es recomendable compartirlas en vez de volver a repetir el código. Para ello podemos utilizar **selectores_combinados**, tal y como hemos visto anteriormente.

```
h1, h2, h3 { font-family: Arial; font-weight: 700; }
```

6. Utilizar selectores descendientes

Es conveniente utilizar **selectores descendientes** siempre que sea posible antes de crear un selector clase o un selector identificador. De esta forma, el código estará más limpio, dispondrá de menos selectores clase e identificador y se comprenderá mucho mejor.

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

7. Utilizar propiedades abreviadas

Siempre que sea posible es recomendable utilizar clases abreviadas para conseguir una reducción de código.

```
/* Propiedades margin-left, margin-right y margin-top */
.nav-menu {margin-left: 5px; margin-right: 5px; margin-top: 5px;}
/* Propiedad abreviada margin */
.nav-menu {margin: 5px 5px 0px 5px;}
```

8. Utilizar nombres descriptivos en los selectores

Mediante la utilización de nombres que permitan averiguar fácilmente a qué elemento le estamos dando estilo comprenderemos el código fácilmente.

```
.nav-button { background: blue; } /* Estilo del botón de la navbar*/
```

9. Evitar utilizar como nombre de un selector una característica visual

Al utilizar en el nombre de un selector una característica visual como el color, el tamaño o la posición, si posteriormente modificamos esa característica, también deberíamos cambiar el nombre del selector. Esto complica mucho el código ya que tendríamos que actualizar todas las referencias a ese selector en el código HTML.

```
/* Selector con nombre que define la característica visual del color */
.menu-red { background: red; }
/* Utilizar mejor: */
.nav-menu { background: red; }
```

10. Probar el diseño en los diferentes navegadores

Para descubrir si se producen errores de visualización en los navegadores lo recomendable es instalarlos todos en el equipo. Algunos de los más utilizados son los siguientes:

5. Chrome
6. Firefox
7. Opera
8. Safari
9. Microsoft Edge

También puedes hacer uso de la aplicación browserling que te permite ver tu desarrollo en varias versiones diferentes de cada navegador.

11. Validar el código CSS

Detectar errores en el código es esencial y se puede hacer fácilmente mediante un validador CSS. El W3C proporciona una herramienta de validación de CSS gratuita para los documentos CSS.

12. Agregar los prefijos de los navegadores en propiedades que no sean estables

Para ahorrar tiempo y facilitarnos la tarea de incluir los prefijos de las propiedades CSS que todavía no son estables podemos hacer uso de la extensión «Autoprefixer» en Visual Studio Code.