

DESARROLLO DE APLICACIONES WEB / MULTIPLATAFORMA

UD 3 Definición de estilos en entornos web

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

JOSÉ HIPÓLITO BOGAS PERONA

Índice

0.- Objetivos.....	3
3.- Introducción.....	3
3.1.- Introducción, evolución y estado actual de CSS.....	4
3.2.- Estructura y sintaxis de CSS.....	5
3.3.- Aplicación de CSS.....	6
3.4.- Prioridades y orden en CSS.....	8
3.5.- Valores y unidades en CSS.....	10
3.5.1.- Tipos de datos.....	10
3.5.2.- Unidades de longitud absoluta.....	11
3.5.3.- Unidades de longitud relativa.....	11
3.5.4.- Colores.....	13
3.5.5.- Imágenes.....	14
3.5.6.- Tipos de letras.....	15
3.6.- Selectores CSS.....	17
3.6.1.- Selectores básicos.....	18
3.6.2.- Agrupación de selectores.....	21
3.6.3.- Combinadores.....	21
3.6.4.- Pseudoclases.....	23
3.6.5.- Pseudoelementos.....	27
3.7.- Propiedades CSS.....	29
3.7.1.- Contenedores y posicionamiento: el modelo de cajas.....	29
3.7.2.- Propiedad float.....	36
3.7.3.- Propiedad position.....	39

UD 3 Definición de estilos en entornos web	MD 8501PR01V15
IES Ágora Cáceres	
REV. 1	Pág. 3/51

3.7.4.- Propiedad overflow y el contenido desbordante.....	42
3.7.5.- Propiedades de texto.....	43
3.7.6.- Propiedades de listas.....	45
3.7.7.- Propiedades de tablas.....	45
3.7.8.- CSS aplicado a formularios.....	46
3.7.9.- CSS aplicado a enlaces.....	48
3.7.10.- Fondos, bordes y contornos.....	48
3.7.11.- Diseño adaptativo con media queries.....	49
3.8.- Bibliografía y webgrafía.....	51

0.- Objetivos

1. Conocer la estructura y la sintaxis de CSS.
2. Aprender cómo se aplica CSS a los documentos.
3. Presentar los tipos utilizados para determinar colores, tipos de letras y unidades de medida.
4. Enumerar y descubrir los diferentes tipos de selectores.
5. Desglosar las propiedades que se pueden modificar desde los estilos CSS.

3.- Introducción

CSS (**Cascading Style Sheet**) es un potente mecanismo para agregar estilo (colores, tipos de letra, espaciados etc) a las páginas web, pero no es un lenguaje de marcas. La pregunta que surge a continuación es obvia: ¿por qué tratar css en este módulo?.

Con el paso del tiempo y las varias versiones, HTML ha dejado de contener información referente a cómo se deben presentar los datos en las páginas web, delegándose esta tarea en las hojas de estilo CSS. Sólo con HTML las presentaciones de las páginas serían extremadamente austeras y carentes de atractivo. CSS está presente en casi todos los documentos HTML existentes, por lo que no se concibe una tecnología sin la otra. Por esta razón es necesario incluir CSS en este módulo, ya que se puede considerar qué forma con HTML un equipo inseparable.

3.1.- Introducción, evolución y estado actual de CSS

CSS son las siglas de *Cascading Style Sheets* (Hojas de estilo en cascada), y es el lenguaje que se utiliza para definir el aspecto de las páginas HTML y XHTML.

Se diseñó para separar los datos (contenidos y los documentos HTML) de las reglas de presentación (contenidos en los documentos CSS), ya que, con anterioridad a su aparición las reglas de presentación formaban parte de las etiquetas HTML.

Esta separación proporciona un enorme versatilidad a la hora de presentar la información, pues un mismo documento HTML puede combinarse con diferentes hojas de estilo CSS para generar distintas presentaciones.

En las siguientes figuras se muestra el mismo documento HTML utilizando distintas hojas de estilo CSS. Las diferencias son evidentes: la tipografía, el color del texto, el color de fondo, la presencia o ausencia de las viñetas de la lista y la alineación del texto. Estos cambios se han introducido utilizando distintas hojas CSS sobre el mismo contenido HTML.

Las tecnologías básicas para crear contenidos Web son tres:

- HTML5
- CSS3
- JavaScript

Figura 3.1. Con CSS se asigna estilo a los documentos HTML.

Las tecnologías básicas para crear contenidos Web son tres:

HTML5
CSS3
JavaScript

Figura 3.2. Un mismo documento HTML puede tener múltiples presentaciones.

Al igual que HTML, es el consorcio W3C la entidad encargada del mantenimiento y la estandarización de CSS. La primera versión CSS vio la luz en 1996 y desde entonces ha habido sucesivas modificaciones hasta llegar a la versión actual, denominada CSS3. No obstante, hay que aclarar algunos aspectos relacionados con las versiones. Hasta la versión 2.1, todas las anteriores se presentaban como una especificación única. Por cuestiones de organización, el W3C decidió que la especificación de la versión 3 se realizaría por módulos, por lo que ya no existe una única versión actual, sino módulos actualizados a la versión 3 y módulos en versión 2.1. Sea como sea, en la jerga técnica se hace referencia a CSS3 como la versión actual de CSS, sin entrar en más detalles.

En la siguiente tabla se muestran las diferentes versiones y su año de publicación.

Tabla 3.1. Evolución de las versiones de CSS

AÑO	VERSIÓN	OBSERVACIONES
1996	CSS1	El W3C ya no mantiene esta versión.
1998	CSS2	El W3C ya no mantiene esta versión.
2011	CSS2.1	Corrige errores de CSS2.
2012	CSS3	Esta versión está dividida en módulos, por lo que no existe una única fecha global de cambio de versión.

«Todas las referencias realizadas en esta unidad HTML afectan de manera implícita a XHTML, ya que el modelo de trabajo en relación con CSS es idéntico en ambos lenguajes.»

3.2.- Estructura y sintaxis de CSS

El funcionamiento de CSS consiste en definir unas reglas de presentación que se van a aplicar a un número indeterminado de elementos del documento HTML, al que están vinculadas. Por lo tanto, se necesitan dos herramientas básicas para crear y aplicar un estilo:

- **Selectores:** son las herramientas que permiten seleccionar el elemento o elementos sobre los que aplicar las reglas.
- **Declaraciones:** son las indicaciones para asignar mediante pares de propiedad valor el aspecto deseado a los elementos determinados por el selector.

Los selectores y las declaraciones se agrupan en **reglas**.

- En el siguiente ejemplo se muestra una regla compuesta por los distintos elementos: Selector **body**. El contenido se aplica a todos los elementos que se encuentren dentro del elemento <body> en la página HTML.
- Las declaraciones que asignan valor a las propiedades **text-align**, **font-family** y **background-color** y **color**, que afectarán respectivamente a la alineación del texto, el tipo de letra, el color de fondo y el color del elemento <body> y de todos los elementos contenidos en éste, siempre que las propiedades sean aplicables a dichos elementos.

```
body {
    text-align: center;
    font-family: 'Roboto', sans-serif;
    background-color: black;
    color: white;
}
```

Los comentarios se delimitan por los símbolos /* y */ y no se procesan por el navegador.

/* Esto es un comentario del número de líneas que necesite */

3.3.- Aplicación de CSS

Para que las reglas CSS tengan efecto sobre los elementos de un documento HTML deben vincularse con éste. Esto se puede realizar de 3 maneras distintas:

1. **Como un documento CSS externo.** Las reglas se almacenan en un documento externo con extensión .css y se establece una relación entre el documento HTML y dicho fichero. La asignación se hace mediante elementos **<link>** asignando «stylesheet» como valor al atributo **rel** y la URI del fichero .css al atributo **href**.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="ejemplo1_dark.css">
    <title>Título del documento</title>
</head>
<body>
    <h1>Las tecnologías básicas para crear contenidos Web son tres:</h1>
    <ul>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
    </ul>
</body>
</html>
```

ejemplo1_dark.css

```
body {
    text-align: center;

    background-color: black;
    color: white;
}

ul {
    list-style: none;
}
```

2. **Como una declaración CSS interna.** Las reglas se declaran en un elemento `<style>` incrustado en el propio documental HTML.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <style>
        body {
            text-align: center;
            background-color: black;
            color: white;
        }
        ul {
            list-style: none;
        }
    </style>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Las tecnologías básicas para crear contenidos Web son tres:</
h1>
    <ul>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
    </ul>
</body>
</html>
```

3. **Cómo una asignación inline** (en el propio documento de las reglas). La regla se declaran en el valor de la propiedad **style** del elemento al que afectan.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body style="text-align: center; background-color: black; color: white">
    <h1>Las tecnologías básicas para crear contenidos Web son tres:</
h1>
    <ul style="list-style: none;">
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
    </ul>
</body>
</html>
```

Las tres alternativas no son exclusivas; se puede añadir una referencia a una o más hojas de estilo externas, incluir un elemento `<style>` y añadir un atributo `style` a un elemento. En ningún caso estas alternativas son incompatibles.

Cada una de las opciones para agregar estilos tiene sus ventajas e inconvenientes, siendo la más versátil y reutilizable la solución de crear un documento externo, ya que se podrá utilizar desde diferentes documentos. No obstante, las otras opciones a veces son elegidas por comodidad o para realizar prototipos rápidos, teniendo siempre en cuenta que las reglas no podrán reutilizarse con comodidad.

Actividad propuesta 3.1 Aplicación de estilo

3.4.- Prioridades y orden en CSS

El número de reglas que se puede aplicar a un documento HTML es ilimitado y estas pueden entrar en conflicto. Pendientes de enumerar y explicar los diferentes tipos de selectores se puede adelantar que dos reglas, una que utiliza el selector `body` y otra que utiliza el selector `h1`, afectarán a los elementos `<h1>`. La pregunta que surge es ¿qué regla va a elegir el navegador en caso de que dos o más apliquen al mismo elemento? La respuesta está en la «C» de CSS.

Antes de explicar cómo se asignan las prioridades hay que saber que pueden existir tres hojas de estilo distintas en una página: **La propia del navegador** que es la primera que se aplica por defecto; **la que proporciona el usuario**, que se asocia al navegador, se aplica por defecto a todas las páginas y es muy útil para personas con discapacidad, y por último, están **las hojas de estilo del diseñador**, que se aplican al final y pueden ser más de una.

Hay tres conceptos fundamentales en la aplicación de los estilos: **cascada, especificidad y herencia**.

La **herencia** provoca que los valores de algunas de las propiedades aplicadas a un elemento contenedor se apliquen también a los elementos contenidos por él. No todas las propiedades que se ven afectadas por este comportamiento, ya que en algunos casos no tendría sentido.

Los estilos CSS se aplican en **cascada**, según las reglas fijas, obteniendo unos resultados predecibles. Los factores que participan en la decisión de que estilo aplicar son los siguientes:

- **Origen e importancia.** Los estilos inherentes al navegador tienen menos prioridad que los elegidos por el usuario y éstos, a su vez, menos que los creados por el diseñador de la página, que son los que se acaban imponiendo en caso de conflicto. Dentro de los estilos proporcionados por el diseñador influye si están

definidos **inline** (mayor prioridad), en un elemento **<style>** (prioridad intermedia) o en un recurso externo (menor prioridad).

- **Nivel de especificidad.** Cuanto más específico mayor prioridad.
- **Orden de aparición.** los últimos estilos en leerse o procesarse se imponen a los existentes.

De manera más detallada la **reglas fundamentales de aplicación de estilos en caso de conflictos** son las siguientes:

- La regla definida con el selector universal (*) se aplica a todos los elementos de la página a los que pueda afectar.
- Si una propiedad se modifica en un único punto, no hay conflicto y se aplica.
- Si una propiedad se modifica en el atributo **style** se impone al intento de modificación de la misma propiedad en el elemento **<style>** o en una hoja CSS externa.
- Si una propiedad se modifica en el elemento **<style>**, se impone al intento de modificación de la misma propiedad en una hoja CSS externa.
- Si dos o más hojas CSS modifican la misma propiedad, se impone la hoja que se ha añadido en último lugar.
- Si dos reglas modifican la misma propiedad del mismo elemento se impone aquella cuyo selector es más específico:
 - El selector con mayor número de **identificadores (id)** será el de mayor prioridad.
 - El selector con mayor número de **clases (class)** o **pseudoclases** será el siguiente en el orden de prioridad.
 - El selector con mayor número de **elementos** o **pseudoelementos** será el siguiente en el orden de prioridad.
- En caso de igualdad en las prioridades se impone el último estilo aplicado.

Estas reglas generales se pueden omitir o modificar mediante una serie de valores comunes a todas las propiedades que tienen efecto sobre cómo se realiza la herencia. Estos valores son:

inherit. Activa la herencia haciendo que la propiedad en la que se está asignando como valor herede la configuración del elemento padre.

initial. Al indicar este valor a una propiedad se consigue que tome el valor de la hoja de estilos por defecto para el elemento al que hace referencia.

unset. Restablece el valor de la propiedad a su valor natural.

revert. Restablece el valor de la propiedad al valor que habría tenido si no hubiese sufrido cambios desde ningún origen.

Además CSS dispone de la declaración **!important**. Este atributo anula las propiedades de un elemento como consecuencia de la aplicación de las reglas de asignación.

La declaración se efectúa inmediatamente después de la asignación del valor a la propiedad y justo antes del carácter ";" que finaliza la asignación. En el siguiente código CSS, se puede observar un ejemplo de aplicación:

```
div {
    color:rgb(9, 149, 234) !important;
}
```

Actividad propuesta 3.2 Prioridades

3.5.- Valores y unidades en CSS

Antes de presentar las posibilidades de las hojas de estilo es necesario conocer qué valores admiten los diferentes parámetros y en qué unidades se expresan.

3.5.1.- Tipos de datos

Los tipos de datos que se utilizan a la hora de configurar un estilo son los que se detallan en la siguiente tabla:

Tabla 3.2. Tipos de datos admitidos por CSS

Tipo	Descripción
Entero	Número entero, ya sea positivo o negativo. Por ejemplo 10 o -3.
Número	Un número decimal. Utiliza el punto como separador decimal. Si no hay decimales, se omite el punto.

Tipo	Descripción
Dimensión	Es un número con una dimensión asociada; deg (grados), s (segundos) o px (pixeles) son algunos ejemplos.
Porcentaje	Una fracción de un total, normalmente referente a la dimensión del padre del elemento al que se está aplicando. Por ejemplo 60% en un font-size representa dicho porcentaje del tamaño del texto del elemento padre. Si, en cambio, se aplica un porcentaje a la propiedad width, se aplica dicho porcentaje sobre el ancho del elemento contenedor.

3.5.2.- Unidades de longitud absoluta

Las unidades de longitud absoluta representan longitudes sin importar en qué contexto se encuentre el componente al que hacen referencia.

Tabla 3.3. Unidades absolutas

Unidad	Nombre
px	Pixel
cm	Centímetro (96 px / 2,54)
mm	Millímetro
Q	Cuarto de milímetro
in	Pulgada (96 px)
pt	Puntos (1/72 de in)
pc	Picas (1/16 de in)

3.5.3.- Unidades de longitud relativa

Las unidades de longitud relativas representan longitudes que tienen diferente valor en función del contexto en el que se encuentran.

Tabla 3.4. Unidades relativas

Unidad	Nombre
em	Tamaño de letra del elemento padre.
ex	Altura de la fuente del elemento.
ch	Ancho del carácter «0» de la fuente del elemento.
rem	Tamaño de letra del elemento raíz.

Unidad	Nombre
lh	Altura de la línea del elemento.
vw	1 % del ancho de la ventana gráfica.
vh	1 % del alto de la ventana gráfica.
vmin	1 % de la dimensión más pequeña de la ventana gráfica.
vmax	1 % de la dimensión más grande de la ventana gráfica.

De este conjunto de posibles unidades, las más utilizadas son **em** y **rem**. Las dimensiones absolutas son más sencillas de aplicar pero los resultados dependen mucho

de la resolución y de la densidad de píxeles de la pantalla, por lo que suele ser más eficaz utilizar unidades relativas.

En el siguiente ejemplo se muestran tres textos con diferentes tamaños de letra: en los dos primeros casos se utiliza **em** como unidad de medida mientras que en el tercer caso se asigna un tamaño porcentual. El tamaño asignado al id **saludoTriplePorcentual** (300)% y al id **saludoTriple** (3 em) tienen el mismo efecto, ya que ambos multiplican por tres el tamaño del tipo de letra del elemento.

```
<div>
    Buenos días normal
    <div id="saludoTriplePorcentual">
        Buenos días porcentual
    </div>
</div>
<div id="saludoDoble">
    Buenos días doble
</div>
<div id="saludoTriple">
    Buenos días triple
</div>
```

```
#saludoDoble {
    font-size: 3em;
}
#saludoTriple {
    font-size: 3em;
}
#saludoTriplePorcentual {
    font-size: 300%;
}
```

Buenos días normal

Buenos días porcentual

Buenos días doble

Buenos días triple

Figura 3.3. En CSS se pueden expresar los tamaños en varios tipos de unidades.

3.5.4.- Colores

Los colores se pueden utilizar para colorear textos, fondos de contenedores o bordes, por ejemplo. CSS permite expresar los colores con varias notaciones distintas:

- Por **nombre**. Indicando el nombre del color de la lista de la recomendación del W3C (CSS Color Module Level 3). Así, el color rojo se asigna con el valor **red**. La lista completa se encuentra en el siguiente enlace: <https://www.w3.org/TR/css-color-3/#html4>.
- Por valor **RGB** (del inglés Red, Green, Blue; «rojo, verde, azul»), Indicando la composición de las tres componentes con valores en el rango 0-255 mediante el formato **RGB**. Por ejemplo, el color rojo se asigna con el valor **RGB(255,0,0)**. Con la variante **RGBA(r,b,g,a)** se añade un componente que indica la transparencia del color (lo que se denomina **canal Alfa**) con un valor decimal comprendido entre 0 (completamente transparente) y el 1 (completamente opaco).
- Por valor **hexadecimal**. Indicando el valor en hexadecimal de las componentes RGB mediante el formato **#valorhexadecimal**. Por ejemplo, el color rojo se asigna con el valor **#ff0000**.
- Por valor **HSL** (del inglés Hue, Saturation, lightness: «matiz, saturación, luminosidad») Indicando los valores correspondientes para cada componente Hue entre 30° y 360°, Saturation entre 0% de 100%; Lightnes entre 0% y 100% en formato **HSL**. Por ejemplo, el color rojo se asigna con el valor **hsl (0, 100%, 50%)**. Con la variante **HSLA(h, s, l, a)** se añade un componente que indica la transparencia del color con un valor decimal comprendido entre 0 (completamente transparente) y 1 (completamente opaco).

A continuación, se muestran dos ejemplos de asignación de color sobre un mismo código HTML expresando dichos colores con diferentes notaciones.

```
<div id="noticia">
    El diario <cite>ABC</cite> redactó la increíble noticia de la siguiente
    manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en
    el Mar de la Tranquilidad."
</div>
```

Notación por valor RGB:

```
#noticia {
    padding: 1em;
    background-color: rgb(200,200,200);
}
```

El diario ABC redactó la increíble noticia de la siguiente manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en el Mar de la Tranquilidad."

Figura 3.4. Mediante RGB se pueden referenciar hasta un total de 16.777.216 colores distintos.

Notación por nombre:

```
#noticia {  
    padding: 1em;  
    background-color: wheat;  
}
```

El diario *ABC* redactó la increíble noticia de la siguiente manera: "Un pie se descuelga por las escalerillas del módulo lunar posado en el Mar de la Trasquilidad."

Figura 3.5. La notación por nombre permite recordar los colores más fácilmente.

3.5.5.- Imágenes

Las referencias a imágenes admiten dos tipos de valores: archivos o degradados.

Los archivos se refencian utilizando la función **url()**. Esta referencia puede ser local o a un recurso externo.

En el siguiente ejemplo se asigna una imagen de fondo a un elemento **<div>**. La imagen de fondo del elemento **<div>** con identificador **huertos** se obtiene de un fichero alojado en la misma carpeta en la que se encuentra el documento CSS.

```
<div id="huertos">  
    <p>Los huertos ecológicos son cada vez más populares gracias a la calidad  
    de sus productos.</p>  
</div>
```

```
#huertos {  
    padding: 1em;  
    color:whiteSmoke;  
    font-weight: bolder;  
    background-image: url("huerta.jpg");  
}
```



Los huertos ecológicos son cada vez más populares gracias a la calidad de sus productos.

Figura 3.6. Las imágenes de fondo son un recurso gráfico muy llamativo.

La misma imagen, podría estar alojada en un servidor web. En ese caso, la referencia en la Hoja de estilo CSS tendrá el siguiente aspecto:

background-image: url("http://MiServidorWeb.es/huerta.jpg");

Un degradado o gradiente de color técnicamente no tiene la consideración de color y no se puede aplicar como tal, sino que se considera una imagen, por lo que su utilización debe realizarse en el contexto adecuado. Para crear una imagen con un gradiente hay que utilizar la palabra **linear-gradient**, **radial-gradient**, **conic-gradient**, **repeating-linear-gradient** o **repeating-radial-gradient** e indicar los colores que utilizar, la dirección, el momento de la transición entre colores, etcétera. Las posibilidades son casi infinitas y permiten a los diseñadores dar rienda suelta a su creatividad.

En el siguiente ejemplo se muestra el elemento **<div>** con un gradiente de color de fondo.

```
<div id="noticia">
    El diario <cite>La Vanguardia</cite> publicó la noticia en portada
    concediéndole la importancia que se merecía: "Los habitantes de la Tierra, en
    vela permanente, contemplan, a través de la Televisión, con asombro y emoción
    como un hombre llamado Neil Armstrong baja por una escalera y pone el pie en
    la Luna".
</div>
```

El diario *La Vanguardia* publicó la noticia en portada concediéndole la importancia que se merecía: "Los habitantes de la Tierra, en vela permanente, contemplan, a través de la Televisión, con asombro y emoción como un hombre llamado Neil Armstrong baja por una escalera y pone el pie en la Luna".

Figura 3.7. Los gradientes son altamente configurables.

3.5.6.- Tipos de letras

En computación se conoce como tipo de letra, tipografía o fuente al conjunto de modelos gráficos que representan cada uno de los caracteres y símbolos representables por el ordenador y que se almacena en un fichero. Existe un número casi ilimitado de tipos de letras creados por los diseñadores gráficos y, como puedes imaginarte, no están todos disponibles en todos los ordenadores.

Para poder disponer de un tipo de letra específico hay que tener el fichero que almacena la información gráfica en el ordenador o, en caso contrario, algún mecanismo para obtener dicha información en el momento de su uso.

Desde el punto de vista del diseño web existen ciertas tipografías que se pueden utilizar de manera segura y general, ya que son comunes a todos los sistemas. Se denominan tipos de letra seguros y la lista no es muy extensa: **Arial**, **Courier New**, **Georgia**, **Times New Roman**, **Trebuchet MS** y **Verdana**. De manera específica CSS define cinco tipos de

letra genéricos cuya representación puede no ser exactamente la misma en todos los sistemas. Estos tipos genéricos son: **serif**, **sans-serif**, **monospace**, **cursive** y **fantasy**.

Este texto está escrito con tipo de letra elegido por defecto por el navegador.

Este texto está escrito con tipo de letra Courier New

Este texto está escrito con tipo de letra monospace

Figura 3.8. Solo algunas tipografías están disponibles en todos los ordenadores.

A partir de este conjunto de opciones básicas de tipos de letras, se abre un universo de múltiples y muy atractivas tipografías creadas por diseñadores y artistas. De forma general, los navegadores admiten formatos de letra diferentes, por lo que hay que especificar alternativas para todos ellos. El formato más común entendido por la mayoría de los navegadores es *The Web Open Font Format* en sus versiones 1 y 2 (extensiones .woff y .woff2). Además, un gran número de navegadores admiten los tipos de letra *True Type Fonts* (extensión .ttf), *Open Type Extension* (extensión .otf) y *SVG Fonts* (extensión .svg).

Se puede referenciar a una fuente no genérica en la Hoja de estilo CSS. Si está instalada en el ordenador, se mostrará correctamente. Si no está disponible se mostrará con el tipo de letra por defecto.

Este texto está escrito con un tipo de letra local

Este texto está escrito con un tipo de letra no disponible

Figura 3.9. Los navegadores tienen un plan alternativo si el tipo de letra elegido no está disponible.

Para evitar los inconvenientes surgidos por no tener instalado el tipo de letra en los ordenadores de los usuarios. CSS permite cargar la tipografía desde un servidor.

La carga de tipografía se hace con la regla @font-face. Tiene la siguiente sintaxis básica:

```
@font-face {  
    font-family: alias-de-la-fuente;  
    src: url("url-de-la-fuente");  
}
```

El siguiente código CSS asigna al alias **fuente-local** una fuente almacenada en el fichero **Rubik-Bold.ttf**. Este fichero estará almacenado en el servidor y lo descargará el cliente cuando sea necesario, garantizando la disponibilidad del tipo de letra.

```
@font-face {  
    font-family: fuente-local;  
    src: url("fonts/Rubik-Bold.ttf");  
}
```

Por supuesto, el fichero, con la tipografía, puede estar almacenado en un servidor distinto del que aloja la página HTML, tal y como se puede ver en el siguiente código de ejemplo.

```
@font-face {  
    font-family: 'Hanalei Fill';  
    src: url("https://fonts.gstatic.com/s/hanaleifill/v9/fc1mPYtobGbfyQsmIaqzPQi8UAja.woff2");  
}
```

Otra alternativa para importar fuentes de letra externas a la página web, consiste en utilizar la regla **@import**. Esta regla permite obtener un recurso a partir de una URL.

En el siguiente código se muestra la regla de carga y el uso de una fuente de letra:

```
@import url('https://fonts.googleapis.com/css2?family=Roboto');  
  
body {  
    font-family: 'Roboto', sans-serif;  
}
```

Debido a que el formato del fichero que contiene el tipo de letra puede no ser compatible con el navegador Es recomendable proporcionar alternativas que incluyan varios formatos distintos.

«No todas las fuentes son gratuitas. Asegúrate de que tienes los derechos adecuados sobre las fuentes que uses en tus páginas y aplicaciones web.»

3.6.- Selectores CSS

Los selectores constituyen el mecanismo que determina sobre qué elementos o elementos se debe aplicar un estilo. Es un sistema muy elaborado y capaz de conseguir un alto nivel de precisión a la hora de terminar sobre qué propiedades de qué elementos se van a aplicar los valores deseados.

Se puede considerar en los selectores como una especie de cribas de diferentes calibres, capaces de permitir o impedir el paso de los elementos en función de su tamaño. En el caso concreto de CSS, el tamaño del elemento no es relevante, pero los elementos de HTML disponen de otros atributos que pueden participar en este proceso de selección; sólo a aquellos elementos que «pasen el filtro» se les aplicará el estilo correspondiente.

Los selectores tienen un nivel de especificidad dispar, desde muy generales hasta muy específicos. Lo mismo ocurre con el nivel de complejidad; hay selectores muy sencillos y fáciles de comprender y otros más elaborados y complejos.

Se pueden agrupar los selectores como básicos, combinadores, pseudoclases y pseudoelementos.

3.6.1.- Selectores básicos

Son los selectores más sencillos de entender y de uso más habitual. Se detallan en la siguiente tabla:

Tabla 3.5. Selectores básicos

Nombre	Selecciona	Sintaxis
Universal	Todos los elementos.	*
De tipo	Todos los elementos de un tipo determinado.	<i>nombreElemento</i>
De clase	Todos los elementos de una determinada clase.	. <i>nombreClase</i>
De identificador	El elemento con el identificador indicado.	# <i>identificador</i>
	Los elementos que tienen un determinado atributo con unas características específicas. = → El atributo tiene un determinado valor. *= → El atributo contiene un determinado valor. ^= → El atributo comienza con un determinado valor. \$= → El atributo termina con un determinado valor. ~= → El atributo contiene un determinado valor en una lista de valores.	[atributo] <i>elemento[atributo]</i> <i>elemento[atributo="cadena"]</i> <i>elemento[atributo*= "cadena"]</i> <i>elemento[atributo^= "cadena"]</i> <i>elemento[atributo\$= "cadena"]</i> <i>elemento[atributo~= "cadena"]</i>
De atributo		

En el siguiente ejemplo se utilizan selectores de tipo universal, de elemento, de clase, de identificador y de atributo.

```
<body>
<h1>RESUMEN ANUAL</h1>
<form action="">
    <input type="text" placeholder="Departamento">
    <input type="number" placeholder="Límite de ventas">
</form>
<p id="introduccion">
    Durante la temporada los resultados han sido satisfactorios...
</p>
<table>
    <tr>
        <th>Ingresos</th>
        <th>Gastos</th>
        <th>Diferencia</th>
    </tr>
    <tr>
        <td>100</td>
        <td>50</td>
        <td class="positivo">50</td>
    </tr>
    <tr>
        <td>200</td>
        <td>190</td>
        <td class="positivo">10</td>
    </tr>
    <tr>
        <td>80</td>
        <td>180</td>
        <td class="negativo">-100</td>
    </tr>
</table>
</body>
```

```

/* Selector universal */
* {
    margin:3px;
}

/* Selector de elemento */
h1 {
    color: rgb(50, 50, 50);
    font-style: italic;
}

/* Selector de identificador */
#introduccion {
    border-style: dotted;
    padding: 1em;
}

/* selectores de clase */
.positivo {
    color:green;
    font-weight: bolder;
}
.negativo {
    color: red;
    font-weight: bolder;
}

/* selectores de atributo */
input[type]{
    background-color: rgb(196, 248, 196);
}
input[type=text]{
    color: rgb(0, 99, 255);
    font-weight: bolder;
}

```

RESUMEN ANUAL

Tecnología

10000

Durante la temporada los resultados han sido satisfactorios..

Ingresos Gastos Diferencia

100	50	50
200	190	10
80	180	-100

Figura 3.11. Conocer los selectores es fundamental para poder aplicar los estilos CSS.

3.6.2.- Agrupación de selectores

En ocasiones se desea aplicar el mismo estilo a varios elementos. Una posible solución consiste en crear un selector distinto para cada uno de ellos con las mismas propiedades y los mismos valores, aunque esta solución puede no ser la mejor.

CSS permite crear reglas con selectores múltiples pudiendo así aplicar las mismas declaraciones a distintos grupos de elementos.

La sintaxis es la siguiente:

elemento, elemento, elemento { propiedad : valor }

Ejemplo de creación de un selector que aplique el mismo estilo a varios elementos:

```
<div>
  <h1>CSS</h1>
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.</p>
  <h2>Introducción</h2>
  <p>Consequuntur veniam, quaerat repellendus facilis laborum magni
    reprehenderit.</p>
  <h2>Breve historia</h2>
  <p>Doloribus fugit, nesciunt sequi pariatur adipisci rem nostrum optio...</p>
</div>
```

```
h1, h2, h3 {
  color: blue;
}
```

CSS

Lorem ipsum dolor sit amet consectetur adipisicing elit.

Introducción

Consequuntur veniam, quaerat repellendus facilis laborum magni
reprehenderit.

Breve historia

Doloribus fugit, nesciunt sequi pariatur adipisci rem nostrum optio...

Figura 3.12. Al agrupar selectores se reduce el tamaño de las hojas de estilo.

3.6.3.- Combinadores

Son selectores en los que se tiene en cuenta la relación entre elementos en la estructura jerárquica del documento.

Tabla 3.6. Combinadores

Nombre	Combinador	Descripción
De hermanos	A ~ B	A y B son hermanos.
De hijos	A > B	B es hijo de A.
De hermanos adyacentes	A + B	A y B son hermanos y B está inmediatamente a continuación de A.
De descendientes	A B	B es descendiente de A, pero no necesariamente es hijo directo.

Vamos a ver un ejemplo. Dado el siguiente código HTML:

```
<div>
  <div>
    <h2>Sistema operativo</h2>
    <p>El sistema operativo es el componente...</p>
  </div>
  <article>
    <p>Un buen antivirus protege al ordenador...</p>
  </article>
</div>
```

Esta regla centra el texto de los elementos **<p>** que sean hijos de un elemento **<article>**.

```
article > p {
  text-align: center;
}
```

Sistema operativo

El sistema operativo es el componente...
Un buen antivirus protege al ordenador...

Figura 3.13. Los combinadores son más complejos de usar, pero proporcionan mayor nivel de precisión.

La siguiente regla centrará el texto de los elementos **<p>** que sean hermanos de un elemento **<h2>**.

```
h2 ~ p {
  text-align: center;
}
```

Sistema operativo

El sistema operativo es el componente...
Un buen antivirus protege al ordenador...

Figura 3.14. Los selectores creados con combinadores posibilitan múltiples alternativas.

La siguiente regla centrará el texto de los elementos **<p>** que sean descendientes de un elemento **<div>**.

```
div p {  
    text-align: center;  
}
```

Sistema operativo

El sistema operativo es el componente...
Un buen antivirus protege al ordenador...

Figura 3.15. Practicamente todas las relaciones entre elementos se pueden expresar con combinadores.

3.6.4.- Pseudoclases

Se añade como palabra clave a continuación un sector convencional, creando un nuevo filtro de selección en función del estado del elemento o de los elementos filtrados por el selector.

La sintaxis es:

selector:pseudoclase { propiedad: valor; }

En la siguiente tabla se muestra una extensa relación de las pseudoclases existentes. Se omiten aquellas que son experimentales y no tienen un soporte generalizado por parte de los navegadores o cuyo uso es poco habitual.

Tabla 3.7. Pseudoclases

Nombre	Estado del elemento o elementos proporcionados por el selector
:active	El elemento ha sido activado por el usuario.
:checked	Afecta a elementos <code><input></code> de tipo <code>radio</code> o <code>checkbox</code> o a elementos <code><option></code> cuando han sido marcados.
:default	Elemento de un formulario marcado como predeterminado.
:disabled	El elemento está deshabilitado.
:empty	El elemento no tiene hijos.
:enabled	El elemento está habilitado.
:first-child	El primer elemento de entre un grupo de elementos hermanos.
:first-of-type	El primer elemento de un tipo de entre un grupo de elementos hermanos.
:focus	El elemento (de un formulario) tiene el foco.
:focus-within	El elemento (de un formulario) o uno de los elementos contenidos tiene el foco.
:hover	El cursor del ratón se encuentra sobre el elemento.
:indeterminate	El estado es indeterminado. Aplica a los elementos <code><input></code> de tipo <code>checkbox</code> y <code>radio</code> y a los elementos <code><progress></code> .
:in-range	El valor de un campo <code><input></code> se encuentra dentro del rango indicado por los atributos <code>min</code> y <code>max</code> .
:invalid	El elemento (de un formulario) es válido.
:lang(<i>id idioma</i>)	El elemento está en un idioma determinado. Por ejemplo :lang(es)
:last-child	El último elemento de un grupo de elementos hermanos.
:last-of-type	El último elemento de un tipo de entre un grupo de elementos hermanos.
:link	El enlace referenciado por el elemento no ha sido visitado.
:not()	Los elementos que no coinciden con una lista de selectores.
:nth-child(<i>n</i>)	El elemento en la posición <i>n</i> de un grupo de elementos hermanos. Se puede utilizar <code>odd</code> para indicar las posiciones pares y <code>even</code> para las posiciones pares en el parámetro <i>n</i> .
:nth-last-child(<i>n</i>)	El elemento en la posición <i>n</i> de un grupo de elementos hermanos, contando desde el final.
:nth-last-of-type(<i>n</i>)	El elemento de un tipo en la posición <i>n</i> de un grupo de elementos hermanos contando desde el final.
:nth-of-type(<i>n</i>)	El elemento de un tipo en la posición <i>n</i> de un grupo de elementos hermanos.
:only-child	El elemento no tiene elementos hermanos.
:only-of-type	El elemento no tiene elementos hermanos de un determinado tipo.

Nombre	Estado del elemento o elementos proporcionados por el selector
:optional	El elemento (de tipo <code><input></code> , <code><select></code> o <code><textarea></code>) no tiene el atributo required .
:out-of-range	El valor de un campo <code><input></code> se encuentra fuera del rango indicado por los atributos min y max .
:read-only	El elemento no es editable.
:read-write	El elemento es editable.
:required	El elemento es obligatorio.
:root	El elemento de más alto nivel del árbol que representa la estructura de la página.
:target	La zona (por ejemplo, un <code><div></code>), que es referenciada desde un enlace interno.
:valid	El elemento (de un formulario) no es válido.
:visited	El enlace referenciado por el elemento ha sido visitado.

Ejemplo: Utilizando pseudoclases, se colorea el primer elemento de la lista en rojo, el último en azul y el elemento que está en la posición tres en verde.

```

<ol>
    <li>Plataformas</li>
    <li>Simulador</li>
    <li>Deportes</li>
    <li>Estrategia</li>
    <li>Aventura gráfica</li>
</ol>

```

```

li {
    font-weight: bolder;
}

li:first-child {
    color: red;
}

li:nth-child(3) {
    color: green;
}

li:last-child {
    color: blue;
}

```

1. Plataformas
2. Simulador
3. Deportes
4. Estrategia
5. Aventura gráfica

Actividad resuelta 3.1 Asignación de estilo CSS

Asignar estilo a los elementos de la tabla que se define en el código HTML proporcionado, prestando especial interés a los selectores utilizados. Los requisitos son los siguientes:

- ▶ Al encabezado y al pie de la se les asigna un color marrón de fondo, un color de texto blanco, y se convierte el texto mayúsculas.
- ▶ A la última columna se le pone texto en negrita.
- ▶ A las filas impares del cuerpo de la tabla se les asigna un color de fondo añil y un color de texto blanco.
- ▶ A las filas pares del cuerpo de la tabla se les asigna un color de fondo blanco y un color de texto añil.

```
<table>
  <thead>
    <tr>
      <td>Conceptos</td>
      <td>Primer semestre</td>
      <td>Segundo semestre</td>
      <td>Total</td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Alimentación</td>
      <td>1500</td><td>1750</td><td>3250</td>
    </tr>
    <tr>
      <td>Tecnología</td>
      <td>1500</td><td>1750</td><td>3250</td>
    </tr>
    <tr>
      <td>Deporte</td>
      <td>1500</td><td>1750</td><td>3250</td>
    </tr>
    <tr>
      <td>Viajes</td>
      <td>1500</td><td>1750</td><td>3250</td>
    </tr>
```

```

        <tr>
            <td>Papelería</td>
            <td>1500</td><td>1750</td><td>3250</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td>Total</td>
            <td>0000</td><td>0000</td><td>0000</td>
        </tr>
    </tfoot>
</table>

```

SOLUCIÓN:

```

thead>tr, tfoot>tr {
    background-color: brown;
    color: white;
    text-transform: uppercase;
}

td:last-child {
    font-weight: bolder;
}

tbody>tr:nth-child(odd) {
    background-color: rgb(28, 76, 150);
    color: white;
}

tbody>tr:nth-child(even) {
    background-color:white;
    color: rgb(28, 76, 150);
}

```

CONCEPTOS	PRIMER SEMESTRE	SEGUNDO SEMESTRE	TOTAL
Alimentación	1500	1750	3250
Tecnología	1500	1750	3250
Deporte	1500	1750	3250
Viajes	1500	1750	3250
Papelería	1500	1750	3250
TOTAL	0000	0000	0000

Actividad propuesta 3.3 Estilar galería de tarjetas de servicios

3.6.5.- Pseudoelementos

Permite seleccionar elementos por reglas que no forman parte de la estructura general del documento HTML. Al igual que la pseudoclases, se añaden a los selectores, pero no para crear un nuevo nivel de filtrado, sino para seleccionar partes más precisas del conjunto de elementos proporcionados por el selector. La separación entre el selector y el

pseudoelemento se suele hacer con dos puntos dobles (::) para distinguirlos de las pseudoclases, pero los navegadores entienden también los dos puntos simples (:).

La sintaxis es:

selector::pseudoelemento { propiedad: valor; }

En la siguiente tabla se muestran los pseudoelementos de uso más extendido.

Tabla 3.8. Pseudoelementos

Nombre	Descripción
::after	Permite añadir contenido con el atributo content después del elemento seleccionado.
::before	Permite añadir contenido con el atributo content antes del elemento seleccionado.
::first-letter	Primera letra del primer bloque de un elemento seleccionado.
::first-line	Primera línea del primer bloque de un elemento seleccionado.
::selection	Permite modificar el estilo de la selección realizada con el ratón.

Un ejemplo, utilizando pseudoelementos:

- ▶ Añadir el texto «SEGUIR LEYENDO...» inmediatamente después del elemento con el id «ElQuijote».
- ▶ Asignar 3em como tamaño de la primera letra del elemento con id «ElQuijote».

```
<p id="ElQuijote">
    En un lugar de La Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo
</p>
```

```
#ElQuijote:after {
    content: "SEGUIR LEYENDO...";
}
#ElQuijote::first-letter {
    font-size: 3em;
}
```

E n un lugar de La Mancha, de cuyo nombre no quiero acordarme, no ha mucho tiempo que vivía un hidalgo SEGUIR LEYENDO...

Actividad propuesta 3.4 Selectores

3.7.- Propiedades CSS

En CSS es posible modificar prácticamente cualquier aspecto del estilo de un elemento de HTML, ya que el conjunto de propiedades es muy extenso y completo .

En esta unidad se tratan aspectos relativos a funcionamiento del modelo de maquetación utilizado en CSS, conocido como <>modelo de cajas></>. Además, se muestran algunas de las propiedades y técnicas más habituales.

Es conveniente recordar que además de las propiedades específicas de cada elemento, existen propiedades comunes que tienen relación con la herencia y la aplicación en cascada de las reglas CSS.

Recuerda las propiedades **inherit**, **initial**, **unset** y **revert** son comunes y afectan a todos los elementos, permitiendo definir su relación con los elementos que los contienen.

3.7.1.- Contenedores y posicionamiento: el modelo de cajas

La maquetación en CSS se basa en el denominado <>modelo de cajas></> según este concepto, todos los elementos se encuentran ubicados en un contenedor rectangular denominado **caja**.

Estas cajas tienen un conjunto de propiedades básicas comunes que afectan directamente a su ubicación en la página y al espacio que ocupan:

Margen exterior. Es el espacio vacío que se agrega al elemento y que los separa del resto de elementos circundantes. Su dimensión se modifica con la propiedad **margin** y sus variantes **margin-top**, **margin-bottom**, **margin-left** y **margin-right**.

Margen interior. Es el espacio vacío que se agrega al elemento y separa del borde a su contenido. Su dimensión se modifica con la propiedad **padding** y sus variantes **padding-top**, **padding-bottom**, **padding-left** y **padding-right**.

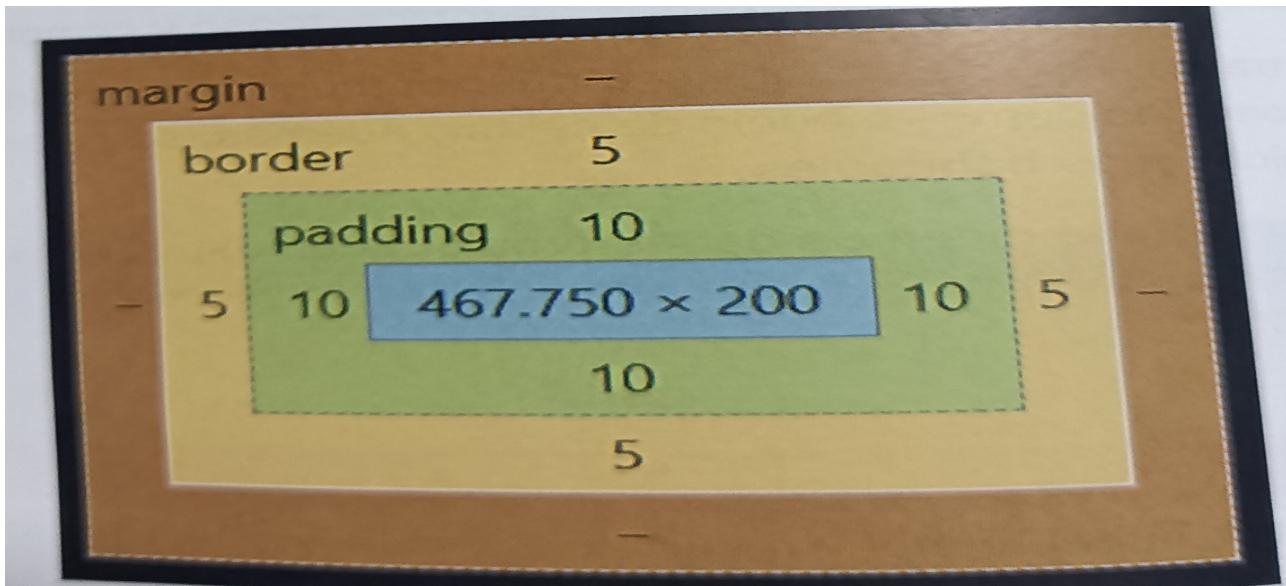
Borde. Es un espacio que puede tener distintos tipos de relleno y que se encuentra entre el margen exterior y el margen interior. Se configura con la propiedad **border** y ocupa espacio, por lo que aumenta el tamaño del elemento. Algunas propiedades para configurar los bordes son **border-radius**, **border-style**, **border-width** y **border-color**.

Contorno. Es similar al borde pero no ocupa espacio ya que se dibuja encima del elemento. Se configura con las propiedades **outline**, **outline-color**, **outline-width** y **outline-style**.

Ancho. Representa el ancho del espacio del contenedor que puede contener elementos. Se modifica con la propiedad **width**.

Alto. Representa el alto del espacio del contenedor que puede contener elementos. Se modifica con la propiedad **height**.

En la siguiente imagen proporcionada por el inspector de Google Chrome en las <>herramientas para desarrolladores>>, se puede observar un ejemplo que de qué espacio ocupan en un elemento cada una de las partes que componen una caja.



Esta figura se basa en los datos del siguiente ejemplo: se aplica a un elemento <div> vacío un estilo en el que se define un ancho porcentual del 50% del espacio disponible, un alto absoluto de 200 píxeles, un margen exterior absoluto de 0 píxeles, un margen interior absoluto de 10 píxeles y un borde de 5 píxeles de ancho.

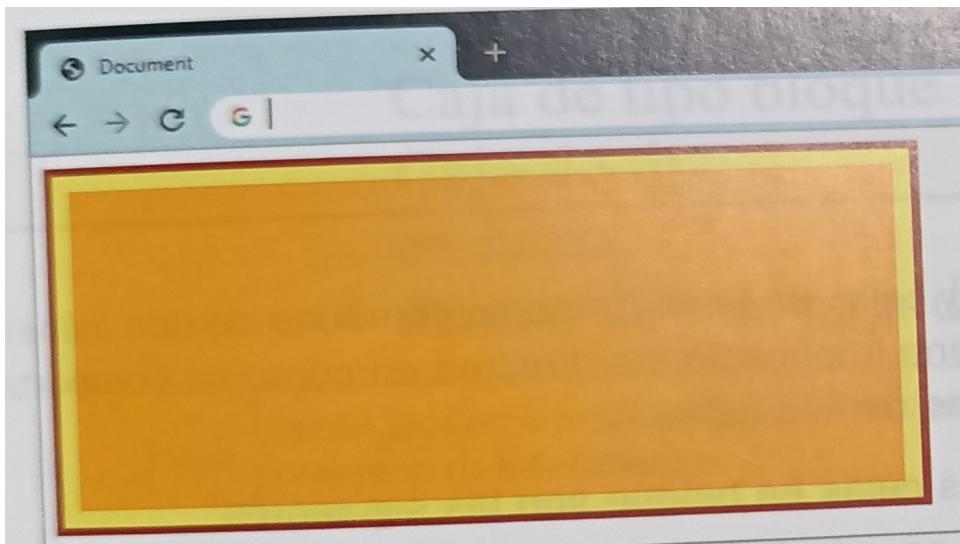
```
<div id="divExterior">
    <div id="divInterior"></div>
</div>

#divInterior{
    background-color: rgb(255, 148, 25);
    width: 100%;
    height: 100%;
}

#divExterior {
    width: 50%;
    height: 200px;
    margin: 0px;
    padding: 10px;
    border-width: 5px;

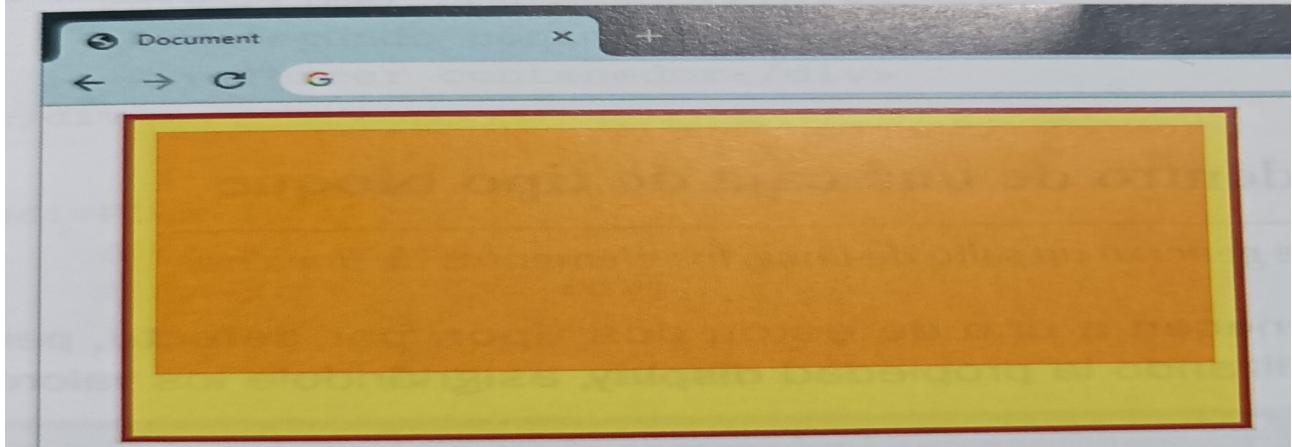
    border-style: solid;
    border-color: brown;
    background-color:rgb(250, 230, 120);
}
```

El resultado se puede observar en la siguiente figura. Pese a que al margen indicado en el estilo no tiene grosor, se aprecia un margen entre el <div> y el navegador: El margen del propio elemento <body>.



Como se ha comentado anteriormente los márgenes izquierdo y derecho se pueden asignar de manera independiente a cada uno de los lados:

```
#divExterior {  
    width: 50%;  
    height: 200px;  
    margin: 0px;  
    margin-left: 30px;  
    padding: 10px;  
    padding-bottom: 50px;  
    border-width: 5px;  
  
    border-style: solid;  
    border-color: brown;  
    background-color: rgb(250, 230, 120);  
}
```



<<Para eliminar los márgenes del navegador y hacer que los contenidos aparezcan pegados a los bordes de éste se puede usar la siguiente regla:

```
body {  
    margin: 0px;  
}
```

Hay dos tipos de cajas: **cajas en bloque** y **cajas en línea** (también llamadas de tipo inline). El que una caja sea de un tipo u otro tiene implicaciones directas en cómo se comporta la caja en relación con la página y con las demás cajas.

Las **características fundamentales** de las **cajas en bloque** son las siguientes:

- ➔ Despues de la caja se realiza un salto de línea.
- ➔ Suele ocupar el 100% del espacio del contenedor en el que se encuentra.
- ➔ Se obliga a los valores de las propiedades **width** y **height**.
- ➔ Los márgenes, bordes y rellenos desplazan al resto de cajas.

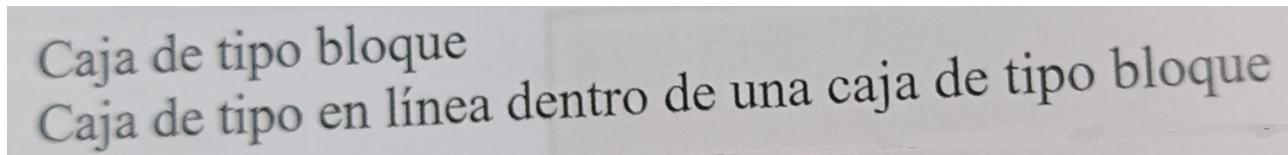
Por su parte las **características principales** de las **cajas en línea** son estas:

- ▶ No generan salto de línea. Por defecto, ocupan el espacio que ocupa su contenido.
- ▶ No se aplican los valores de las propiedades **width** y **height**.
- ▶ Los márgenes, bordes y rellenos horizontales desplazan al resto de cajas en línea.
- ▶ Los márgenes bordes y rellenos verticales no desplazan al resto de cajas en línea.

El siguiente código HTML contiene ejemplos de cajas en línea (elemento **** y en bloque elemento **<div>**).

```
<div>Caja de tipo bloque</div>  
<div>Caja de tipo <span>en línea</span> dentro de una caja de tipo bloque</div>
```

Como se puede observar en la siguiente figura, el primer elemento **<div>** provoca un salto de línea mientras que el elemento **** no lo hace.



Todos los elementos de HTML pertenecen a uno de estos dos tipos de por defecto, pero esta propiedad se puede modificar utilizando la propiedad **display**, asignándole los valores **inline** o **block** según se deseé.

Por ejemplo asignando el código HTML anterior a una modificación de la propiedad **display**, el elemento comenzará a comportarse como un elemento de tipo caja.

```
span {  
    display: block;  
}
```

Caja de tipo bloque
Caja de tipo
en línea
dentro de una caja de tipo bloque

La relación de valores de la propiedad **display** es muy extensa, permitiendo incluso combinarlos. Se divide en aquellos orientados a la visualización **interna** (cómo se distribuyen los componentes **dentro** de la caja) y los orientados a la visualización **externa** (cómo se comporta la caja en relación con los elementos que la rodean). Además, existe el valor de especial **none** que desactiva todas las modificaciones de la propiedad **display** que afecten al elemento en cuestión y a todos los descendientes.

Los valores más utilizados para el tipo de visualización interna son los siguientes:

flex. El elemento se comporta como un elemento de bloque y de acuerdo con el modelo denominado **flexbox**. Mediante la propiedad de **flex-direction** se indica en qué dirección se desea que <<fluya>> el contenido del contenedor. A un contenedor con la propiedad **display** con valor **flex** se le denomina **contenedor flex**.

grid. El elemento se comporta como un elemento de bloque y de acuerdo con el modelo de cuadrícula.

Ejemplos de aplicación del tipo de **display** interno **flex** con diferentes valores para la propiedad **flex-direction**:

```
<div id="divFlex">  
    <div>Primer contenedor</div>  
    <div>Segundo contenedor</div>  
    <div>Tercer contenedor</div>  
</div>  
  
#divFlex {  
    display: flex;  
    flex-direction: row;  
}
```

Primer contenedor Segundo contenedor Tercer contenedor

```
#divFlex {  
    display: flex;  
    flex-direction: row-reverse;  
}
```

Tercer contenedor Segundo contenedor Primer contenedor

Figura 3.25. Con flexbox, se puede invertir el orden natural de los elementos.

```
#divFlex {  
    display: flex;  
    flex-direction: column;  
}
```

Primer contenedor
Segundo contenedor
Tercer contenedor

Figura 3.26. Flexbox trabaja en los ejes horizontal y vertical.

```
#divFlex {  
    display: flex;  
    flex-direction: column-reverse;  
}
```

Tercer contenedor
Segundo contenedor
Primer contenedor

Figura 3.27. Flexbox proporciona la misma flexibilidad en vertical que en horizontal.

<<El módulo de caja flexible o *flexbox* es un modelo de diseño que proporciona un gran una gran versatilidad En el siguiente lapso se puede obtener información detallada sobre este módulo CSS3:

[>>](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout)

Ejemplos de aplicación del tipo de **display** interno **grid** con diferentes valores para la propiedad **grid-template-columns**:

```
<div id="divGrid">
    <div>Primer contenedor</div>
    <div>Segundo contenedor</div>
    <div>Tercer contenedor</div>
    <div>Cuarto contenedor</div>
    <div>Quinto contenedor</div>
    <div>Sexto contenedor</div>
</div>

#divGrid {
    display:grid;
    grid-template-columns: auto auto auto;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

Figura 3.28. El tipo de display grid distribuye los elementos como una tabla.

```
#divGrid {
    display:grid;
    grid-template-columns: 100px 100px 100px;
}
```

Primer contenedor	Segundo contenedor	Tercer contenedor
Cuarto contenedor	Quinto contenedor	Sexto contenedor

Figura 3.29. El tipo grid permite controlar el ancho de los elementos.

```
#divGrid {
    display:grid;
    grid-template-columns: 70% 30%;
```

Primer contenedor	Segundo contenedor
Tercer contenedor	Cuarto contenedor
Quinto contenedor	Sexto contenedor

Algunos de los valores más utilizados para el tipo de visualización externa son los siguientes:

block. El elemento se comporta como un elemento de bloque.

inline. El elemento se comporta como un elemento en línea.

<<Para profundizar en el conocimiento del parámetro display se pueden utilizar los siguientes enlaces:

W3C: <https://www.w3.org/TR/css-display-3/#the-display-properties>

MDN (Mozilla Development Network):

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

3.7.2.- Propiedad float

Por defecto, los elementos de bloque tienen un comportamiento que genera grandes limitaciones a la hora de maquetar las páginas web. Por suerte, este comportamiento se puede modificar mediante los estilos CSS.

Los elementos de bloque que habitualmente se utilizan para maquetar son <header>, <footer>, <aside>, <nav>, <article>, <section> y <div>, siendo este último el más genérico ya que no tiene información semántica asociada. Es importante conocer el comportamiento por defecto de estos elementos y saber realizar las modificaciones adecuadas para que se distribuyan de la forma correcta.

Utilizando el elemento <div> como referencia, en esta sección se muestran algunas técnicas relacionadas con la maquetación en las que se utiliza la propiedad **float** de los elementos de bloque.

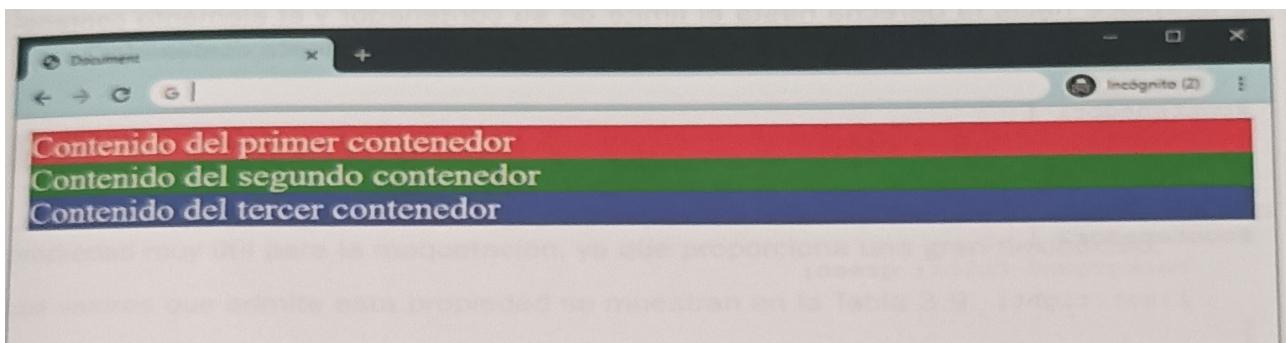
Se puede ilustrar el comportamiento por defecto de los elementos de bloque con un sencillo ejemplo. La siguientes reglas asignan colores distintos a tres identificadores:

```
.contenedor {  
    color: white;  
}  
  
#contenedor1 {  
    color: red;  
}  
  
#contenedor2 {  
    color: green;  
}
```

```
#contenedor3 {  
    color: blue;  
}
```

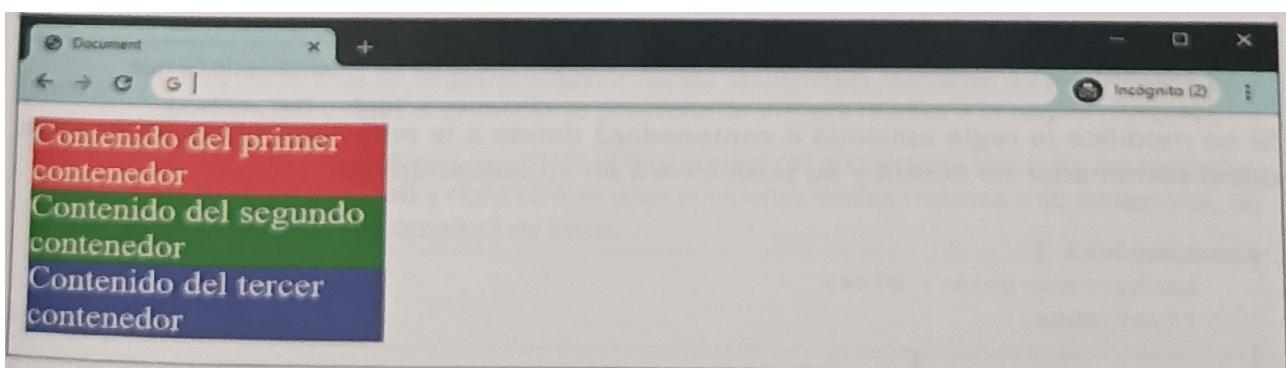
Los elementos de bloque, por defecto, ocupan todo el ancho del contenedor en el que se encuentran y se colocan unos debajo de otros, ya que insertan un salto de línea.

```
<div id="contenedor1" class="contenedor">Contenido del primer contenedor</div>  
<div id="contenedor2" class="contenedor">Contenido del segundo contenedor</div>  
<div id="contenedor3" class="contenedor">Contenido del tercer contenedor</div>
```



Se les asigna un ancho, ya sea en valores absolutos o relativos, los elementos de bloque siguen colocándose de forma vertical.

```
.contenedor {  
    color: white;  
    width: 30%;  
}
```



Este comportamiento por defecto se puede modificar mediante la propiedad **float**. Esta propiedad posiciona el elemento al lado izquierdo o derecho del contenedor en el que se encuentra. Los principales valores que admite son:

left. <<empuja>> el elemento hacia el lado izquierdo de su contenedor.

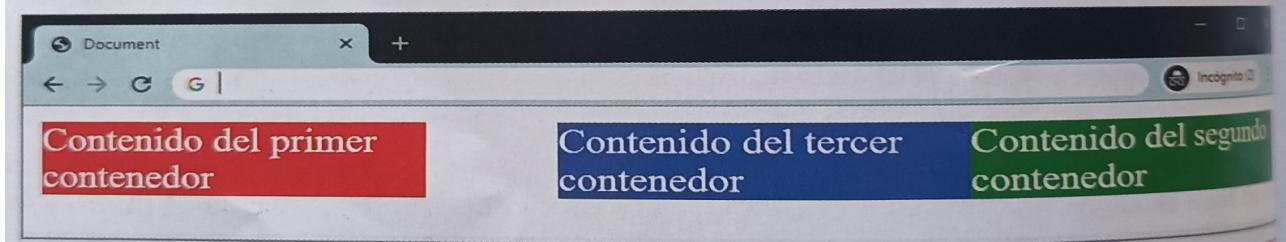
right. <<empuja>> el elemento hacia el lado derecho de su contenedor.

none. No se <<empuja>> al elemento. Es el valor por defecto.

inherit. hereda el valor de la propiedad float del contenedor padre.

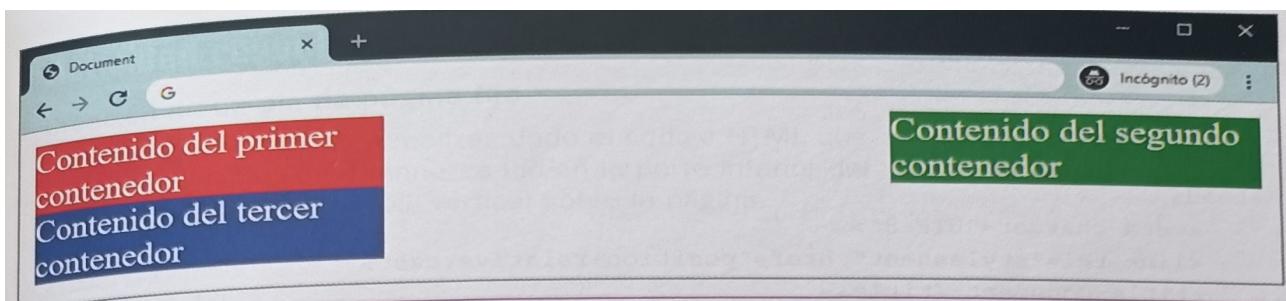
Por ejemplo, modificando el estilo se puede forzar a que el elemento **contenedor1** se desplace hacia la izquierda hasta el límite de su contenedor, el elemento **contenedor2** se desplace hacia la derecha hasta el límite de su contenedor y el elemento **contenedor3** se desplace hacia la derecha hasta encontrarse con elemento **contenedor2**.

```
#contenedor1 {  
    background-color: red;  
    float:left;  
}  
#contenedor2 {  
    background-color: green;  
    float:right;  
}  
#contenedor3 {  
    background-color: blue;  
    float:right;  
}
```



Si se modifica la regla aplicada a contenedor 3 dándole a la propiedad float el valor none dicho contenedor no flotará y se posicionará en su ubicación natural:

```
#contenedor3 {  
    background-color: blue;  
    float:none;  
}
```



<<Se puede centrar un elemento de tipo bloque fijándole un ancho y estableciendo **auto** como valor real del atributo **margin**.>>

3.7.3.- Propiedad position

La propiedad **position** determina cómo se posiciona un elemento en el documento. Es una propiedad muy útil para la maquetación, ya que proporciona una gran flexibilidad.

Los valores que admite esta prioridad se muestran en la siguiente tabla:

Tabla 3.9. Valores admitidos por la propiedad position

Valor	Descripción
static	Es el valor por defecto. El elemento se posiciona siguiendo el flujo normal del documento. No le afectan las propiedades top , bottom , left , right y z-index .
relative	El elemento se posiciona siguiendo el flujo normal del documento, pero se puede desplazar de forma relativa a su posición por defecto utilizando las propiedades top , bottom , left y right . El desplazamiento no desplaza a los demás elementos, por lo que puede proporcionar apilamiento que deberá organizarse utilizando la propiedad z-index .
absolute	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades top , bottom , left y right el elemento se posiciona de manera relativa a su ancestro. Si no tiene ancestro, se posiciona de manera relativa a la página completa.
fixed	El elemento deja de seguir el flujo normal del documento. Mediante las propiedades top , bottom , left y right el elemento se posiciona de manera relativa a la página completa.
sticky	El elemento se posiciona siguiendo el flujo normal del documento. Mediante las propiedades top , bottom , left y right se fijan unas posiciones límites relativas a su contenedor, de tal manera que no excederá de estas.

Ejemplo:

A continuación, se hace uso del posicionamiento relativo para centrar un <<div>> dentro de otro.

El <<div>> principal (el que tiene un atributo id con valor <<div>>1) ocupa todo el ancho disponible y tiene fijado en una altura fijada en píxeles. El tamaño y la posición del id<<div>> interior (el que tiene el atributo y de con valor <<div2>>) se declara relativo al elemento dentro del que se encuentra: el ancho es el 50% del espacio de su contenedor; la posición es de 10 píxeles desde la parte superior del contenedor y de un 25% del ancho del contenedor desde el lado izquierdo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8" >
    <link rel="stylesheet" href="position-relative.css">
    <title>Document</title>
</head>
<body>
    <div id="div1">
        <div id="div2">Texto del div con posición relativa</div>
    </div>
</body>
</html>

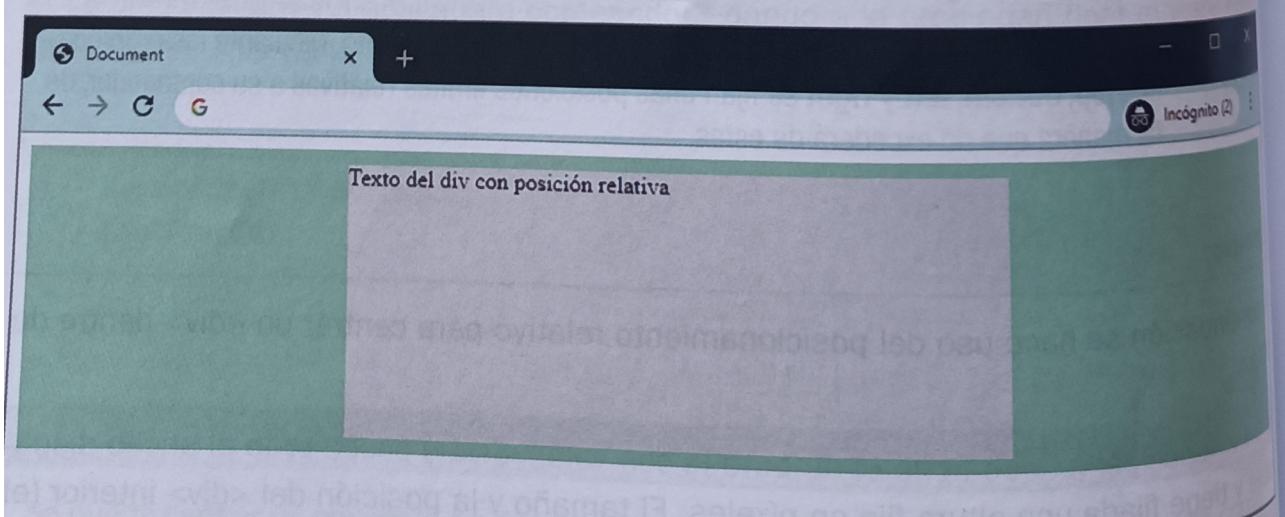
#div1 {
    width: 100%;
    height: 200px;
    background-color: aquamarine;

    position: static;
}

#div2 {
    width: 50%;
    height: 180px;
    background-color: lightgray;

    position: relative;
    top: 10px;
    left: 25%;

}
```



Actividad resuelta 3.2

Creación de un pie de página fijo

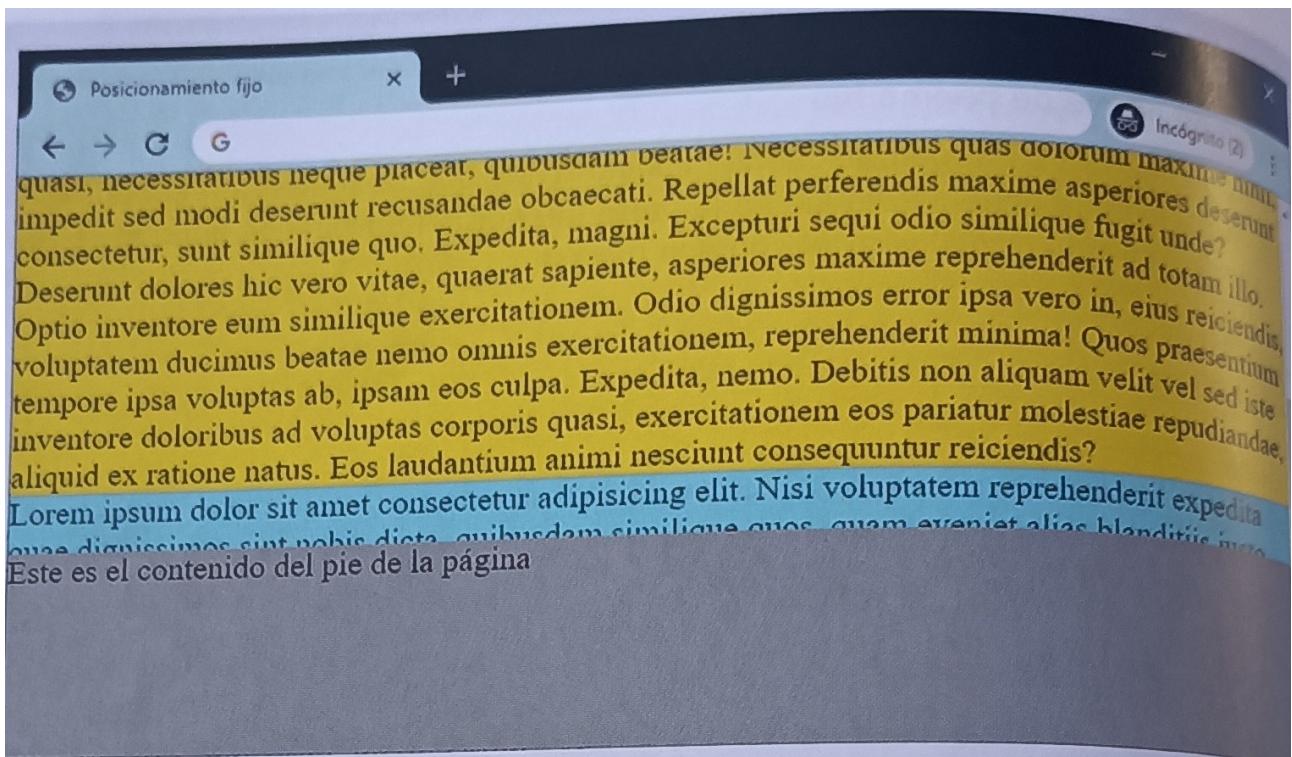
Crear las reglas CSS para realizar, dado el código HTML que se muestra a continuación, un pie de página que permanezca fijo en la parte inferior del navegador, aunque se realice un desplazamiento (scroll) vertical sobre la página.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="position-fixed.css">
    <title>Posicionamiento fijo</title>
</head>
<body>
    <div id="div1">
        <!-- CONTENIDO -->
    </div>
    <div id="div2">
        <!-- CONTENIDO -->
    </div>
    <footer id="pie-documento">
        Este es el contenido del pie de la página
    </footer>
</body>
</html>
```

Solución

El elemento `<footer>` debe tener posicionamiento de tipo **fixed** y estar a 0 píxeles de la parte inferior del navegador (**bottom:0px**). El resultado es un contenedor que está «pegado» al extremo inferior de la página independientemente de la posición del resto de elementos.

```
body {
    margin:0px;
}
#div1 {
    width: 100%;
    background-color: rgb(255, 217, 0);
}
#div2 {
    width: 100%;
    background-color:rgb(0, 255, 255);
}
#pie-documento {
    width: 100%;
    height: 100px;
    background-color: rgb(150,150,150);
    position: fixed;
    bottom: 0px;
```



3.7.4.- Propiedad overflow y el contenido desbordante

En ocasiones, el contenido de un contenedor excede el espacio que tiene asignado. Esto se produce porque a un elemento contenedor (por ejemplo un elemento <div>) se le ha asignado una dimensión y ya no se ajusta al contenido tal y como hace por defecto.

Por defecto, cuando esto sucede en el contenido desborda al contenedor. Es decir, sobrepasa sus límites, pero no los modifica, provocando habitualmente un resultado erróneo.

Existen alternativas a este comportamiento por defecto. Se especifican mediante la propiedad **overflow**, que afecta al contenedor si el contenido lo desborda.

Los valores de la propiedad **overflow** se detallan en la siguiente tabla:

Tabla 3.10. Valores admitidos por la propiedad **overflow**

Valor	Descripción
visible	Es el valor por defecto. Provoca que el contenido desborde el espacio del contenedor consiguiendo así que sea visible por completo.
hidden	Oculta el contenido que desborda al contenedor.
scroll	Aggrega unas barras de desplazamiento (<i>scroll</i>) permitiendo así consultar todo el contenido sin que este exceda de los límites del contenedor.
auto	Se comporta igual que el valor scroll pero solo si el contenido excede del tamaño del contenedor.

Por último, señalar que existen propiedades específicas para determinar el comportamiento del contenido desbordante en los ejes X e Y. Son las propiedades **overflow-x** y **overflow-y**, y se pueden utilizar conjuntamente.

Actividad propuesta 3.5 Posicionamiento de elementos

3.7.5.- Propiedades de texto

Los documentos HTML suelen estar formados, principalmente por textos. La noticia de un periódico, las características técnicas de un producto o los ingredientes de una pizza son textos y éstos deben estar presentados de forma correcta.

Las propiedades de texto permite especificar prácticamente cualquier característica relacionada con los textos, desde el tipo de fuente hasta el espaciado entre entre palabras.

En la siguiente tabla se detallan las propiedades más importantes, su descripción e indicaciones sobre los valores que pueden tomar.

Tabla 3.11. Propiedades aplicables a textos

Propiedad	Descripción	Valores
color	Determina el color del texto.	Cualquier valor válido aceptado como color.
font-family	Determina el tipo de letra que se va a utilizar. Si no puede usar el indicado, aplica el tipo de letra por defecto.	Cualquier valor válido aceptado como tipo de letra.
font-size	Determina el tamaño del texto.	<i>medium, xx-small, x-small, small, large, x-large, xx-large, smaller, larger,</i> un tamaño indicado en alguna de las unidades aceptadas y un porcentaje sobre el tamaño del texto del contenedor.
font-style	Determina el estilo del texto.	<i>normal, italic y oblique.</i>
font-weight	Determina el grosor del trazo de los textos.	<i>normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800 y 900.</i>
text-transform	Transforma el texto.	<i>capitalize, uppercase y lowercase.</i>
text-decoration	Agrega una decoración al texto en forma de línea.	Un valor de las propiedades <i>text-decoration-line, text-decoration-style</i> y <i>text-decoration-color</i> .
text-decoration-line	Dibuja una línea horizontal sobre el texto.	<i>underline, overline y line-through.</i>
text-decoration-style	Determina el tipo de línea (solo en el caso de que haya una propiedad <i>text-decoration-line</i>).	<i>none, underline, overline, line-through</i> combinaciones.
text-decoration-color	Determina el color de la línea (solo en el caso de que haya una propiedad <i>text-decoration-line</i>).	Un color válido.
text-shadow	Añade una sombra al texto.	Un color válido y una distancia en los ejes x e y indicada en cualquiera de las unidades válidas en CSS.
text-align	Determina la alineación horizontal del texto.	<i>left, right, center y justify.</i>
line-height	Determina la altura de la línea.	<i>normal, un número entero sin unidades, un número expresado en cualquiera de las unidades válidas en CSS.</i>
letter-spacing	Determina el espacio entre las letras.	<i>normal, un número expresado en cualquiera de las unidades válidas en CSS.</i>
word-spacing	Determina el espacio entre las palabras.	<i>normal, un número expresado en cualquiera de las unidades válidas en CSS.</i>

3.7.6.- Propiedades de listas

Las listas son uno de los elementos de uso más habitual en HTML. Es interesante por lo tanto conocer las propiedades más importantes.

Tabla 3.12. Propiedades aplicables a listas		
Propiedad	Descripción	Valores más frecuentes
list-style	Permite definir las propiedades <i>list-style-type</i> , <i>list-style-position</i> y <i>list-style-image</i> en una única línea.	Los valores de las propiedades separados por espacios.
list-style-type	Permite definir la apariencia de la lista en lo referente a la viñeta que precede a cada elemento.	<i>disc</i> , <i>circle</i> , <i>square</i> , <i>decimal</i> , <i>lower-roman</i> , <i>upper-roman</i> , <i>lower-greek</i> , <i>lower-latin</i> , <i>upper-latin</i> y <i>none</i> .
list-style-position	Determina dónde se colocarán las viñetas respecto a la posición de la lista.	<i>inside</i> y <i>outside</i> .
list-style-image	Asigna una imagen almacenada en una URL como viñeta.	<code>url("url-de-la-imagen");</code>

3.7.7.- Propiedades de tablas

Al igual que las listas, las tablas son elementos que se utilizan frecuentemente en la maquetación de páginas HTML. Sus propiedades se recogen en la siguiente tabla:

Tabla 3.13. Propiedades aplicables a tablas		
Propiedades	Descripción	Algunos valores posibles
border-collapse	Determina si los bordes de las celdas de la tabla están fusionados o no.	<i>collapse</i> y <i>separate</i> .
border-spacing	Determina la distancia entre bordes adyacentes.	Distancia utilizando las unidades de CSS.
caption-side	Determina dónde aparecerá el texto contenido en el elemento <code><caption></code> .	<i>top</i> y <i>bottom</i> .
empty-cells	Indica si las celdas vacías deben mostrarse o no.	<i>show</i> y <i>hide</i> .
table-layout	Determina el algoritmo que utilizar para diseñar celdas, filas y columnas.	<i>auto</i> y <i>fixed</i> .
vertical-align	Determina la alineación vertical del contenido de las celdas de una tabla.	<i>middle</i> , <i>top</i> y <i>bottom</i> .

3.7.8.- CSS aplicado a formularios

La gran mayoría de los estilos existentes se pueden aplicar a los formularios, por lo que a nivel teórico la aportación de esta sección es casi anecdótica. Desde el punto de vista práctico, diseñar un formulario visualmente atractivo requiere utilizar tipos de letras, contenedores que contengan y ubican los elementos, márgenes internos y externos para espaciar contenidos, contornos o estados de los elementos. Para ilustrar algunas de estas técnicas en la siguiente actividad resuelta se aplican, por medio de CSS, varias soluciones de diseño sencillas a la vez que efectivas.

Actividad resuelta 3.3

Aplicación de estilo a un formulario de registro

Crear un formulario que cumpla los siguientes requisitos:

- El formulario aparecerá centrado en la pantalla.
- Se utiliza un tipo de letra diferente del que aparece por defecto.
- Se solicitan los siguientes campos:
 - Nombre. Tipo texto.
 - Correo electrónico. Tipo email.
 - Contraseña y confirmación de contraseña. Tipo contraseña.
- Los campos tendrán la etiqueta descriptiva justo encima.
- Al poner el foco sobre un campo de entrada se cambiará el color del borde.
- Los campos de entrada y el botón de aceptación tendrán las esquinas redondeadas.
- El botón de aceptación estará coloreado.

El código HTML correspondiente al formulario es:

```
<div id="div-formulario">
    <form action="registro.php" method="post">
        <h1>Crear cuenta</h1>
        <div>
            <label for="nombre">Nombre</label>
            <input type="text" id="nombre" name="nombre" class="entrada borde-redondo">
        </div>
        <div>
            <label for="correo">Correo electrónico</label>
            <input type="email" id="correo" name="correo" class="entrada borde-redondo">
        </div>
        <div>
            <label for="contrasenya">Contraseña</label>
            <input type="password" id="contrasenya" name="contrasenya" class="entrada borde-redondo">
        </div>
        <div>
            <label for="contrasenya-rep">Confirma tu contraseña</label>
            <input type="password" id="contrasenya-rep" name="contrasenya-rep" class="entrada borde-redondo" >
        </div>
```

```

<div>
    <input type="submit" value="Crear tu cuenta" id="boton-aceptar"
class="borde-redondo">
</div>
</form>
</div>

```

Solución

```
@import url('https://fonts.googleapis.com/css2?family=Roboto');
```

```

body {
    font-family: 'Roboto', sans-serif;
}
h1 {
    margin-top:5px;
}
label {
    display: block;
    font-weight: bold;
    margin-bottom: 0.2em;
}
#div-formulario {
    width:350px;
    margin:auto;
}
#boton-aceptar {
    margin-top: 30px;
    width: 100%;
    padding: 12px 3px 12px 3px;
    background-color: #ed904;
}
.entrada {
    margin:0px;
    width: 97%;
    padding: 7px 3px 7px 3px;
    margin-bottom: 1em;
}
.entrada:focus {
    outline: none;
    border-color: #ca5f07;
}
.borde-redondo {
    border: 1px; border-style: solid;
    border-color: #aaaaaa;
    border-radius: 3px;
}

```

Representación obtenida del formulario:

Crear cuenta

Nombre

Correo electrónico

Contraseña

Confirma tu contraseña

3.7.9.- CSS aplicado a enlaces

Los enlaces son los elementos que convierten un documento en hipertextual. Gracias a los enlaces se puede navegar por la web, saltando de recursos en recurso.

El texto de los enlaces, por defecto, se muestra subrayado (utiliza de manera interna la propiedad `text-decoration`) y en color azul cuando no ha sido empleado para navegar a la URL de destino. En el momento de pulsarlo, el color por defecto es rojo. Una vez utilizado, el texto se muestra en color púrpura.

Hay un grupo de cambios de estilo sobre los enlaces que son los mismos que se pueden realizar sobre un texto: El tamaño del texto o el tipo de letra por ejemplo.

Los cambios específicos referentes al enlace se realizan sobre los diferentes estados por los que éste puede pasar. Mediante selectores que incluyan pseudoclases se puede hacer referencia a dichos estados para asignar cambios en la apariencia del enlace.

Tabla 3.14. Selectores relativos a los estados por los que puede pasar un enlace

Selector	Estado
a:link	Es el estado por defecto de un enlace.
a:visited	Se activa cuando el enlace ha sido activado. Sigue estar deshabilitado en los navegadores modernos por razones de seguridad.
a:hover	Se activa cuando se pasa el ratón por encima del enlace.
a:focus	Se activa cuando el foco se posiciona en el enlace.
a:active	Se activa en el momento de la activación del enlace.

3.7.10.- Fondos, bordes y contornos

Existe una importante número de propiedades para caracterizar el fondo, los bordes y los contornos de los elementos. Se pueden aplicar a todo tipo de elementos: ítems de una lista (``), celdas de una tabla (`<td>`), contenedores de bloque (`<div>`, `<footer>`, `<article>`, etc.), contenedores de línea (``), etcétera.

Las propiedades relativas al fondo comienzan por el prefijo **background** y permiten determinar un color o una imagen. En caso de ser una imagen, existe un buen número de propiedades con características relativas a cómo se deben presentar.

Las propiedades relativas al borde de los elementos comienzan por el prefijo **border** y permiten configurar el color, el tipo, el grosor, redondear las esquinas y un largo etcétera.

Al igual que los bordes, se pueden definir contornos (cuya diferencia fundamental es que estos no ocupan espacio) utilizando las propiedades que comienzan por **outline**.

Si deseas profundizar más en los atributos relativos al fondo, los bordes y los contornos puedes hacerlo en los siguientes enlaces:

https://developer.mozilla.org.es/docs/Web/CSS/CSS_Background_and_Borders

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

3.7.11.- Diseño adaptativo con media queries

En la actualidad la diversidad de los tamaños y proporciones de dispositivos es mucho mayor: pequeñas pantallas de teléfono móvil y grandes pantallas de ordenador son medios habituales para acceder a las páginas web.

Esta diversidad requiere de técnicas flexibles que permiten adaptar las reglas de la maquetación con CSS al tamaño o distribución del dispositivo. A estas técnicas se les conoce como el nombre de **diseño adaptativo o responsive**.

El diseño adaptativo se puede conseguir utilizando media queries, que es un mecanismo para asignar propiedades y valores CSS distintos en función del tipo de dispositivo (pantallas, impresoras o sintetizadores de voz) y de sus características, principalmente el tamaño.

Las media queries son, en resumen, un conjunto de reglas que determinan qué valores asignar a determinadas propiedades. El estándar es complejo, pero se puede utilizar a un nivel sencillo y obtener unos resultados más que interesantes.

La manera más sencilla de aplicarlas media queries es a través del elemento <link>, asignando diferentes hojas de estilo en función del tamaño de la página.

En el siguiente código se muestra un ejemplo en el que se asigna una y otra hoja de estilo CSS en función del ancho del medio en el que se está visualizando el documento: Se aplica estilo1.css si el ancho máximo es de 1000 píxeles, o estilo2.css si el ancho del medio tiene un mínimo de 1001 píxeles.

```
<link rel="stylesheet" href="estilo1.css" media="(max-width:1000px)">
```

```
<link rel="stylesheet" href="estilo2.css" media="(min-width:1001px)">
```

Esta sección puede resultar fácil y rápida, pero supone tener duplicado (al menos) la hoja de estilos y esto no suele ser una buena decisión. Para evitarlo existe alternativa de definirlas en función del medio que afecten a una selección.

```
@media (max-width:1000px) {
```

```
    #parrafo {
```

```
        color: springgreen;
```

```
}
```

```
}
```

```
@media (min-width:1001px) {
```

```
    #parrafo {
```

```
        color: rgb(171, 15, 199);
```

```
    }
```

```
}
```

Los parámetros que pueden participar en las media queries son muchos y no se van a detallar en este módulo, pero es conveniente saber que dichos parámetros pueden hacer referencia al tipo de dispositivo (pantalla, impresora, sintetizadores de voz, etcétera), el número de colores que soporta el dispositivo, las ratios o proporciones de aspecto, el alto y ancho del dispositivo, la orientación o la resolución. Además, dispone de la posibilidad de incluir operadores lógicos para crear condiciones más complejas.

Como ejemplo, las siguientes líneas de una hoja de estilo CSS muestran la asignación de diferentes valores para los mismos atributos de selector (body) en función de la orientación del dispositivo que se está utilizando, algo especialmente útil cuando se accede a la página web desde un dispositivo móvil.

```
@media (orientation: portrait) {
```

```
    body {
```

```
        font-size: 18px;
```

```
        background-color: #e74c4c;
```

```
    }
```

```
}
```

```
@media (orientation: landscape) {
```

```
    body {
```

```
        font-size: 20px;
```

```
        background-color: #5a9556;
```

```
    }
```

```
}
```

3.8.- Bibliografía y webgrafía

Bibliografía:

Editorial Paraninfo. Lenguajes de marcas y sistemas de gestión de la información. Fernando Paniagua Martín. 2025

Editorial Garceta. Lenguajes de marcas y sistemas de gestión de la información 2^a edición. M^a Isabel Jiménez Cembreras. 2025

Webgrafía:

W3.CSS Tutorial: <https://www.w3schools.com/w3css/default.asp>

Aprende CSS en 15 Minutos: https://www.youtube.com/watch?v=3yM5uXp-T_0

Aprende CSS Grid Layout en 15 Minutos: <https://www.youtube.com/watch?v=-kgGATnsPbs>

Aprende CSS Flexbox en 15 Minutos: <https://www.youtube.com/watch?v=QFXSgGg-HWo>

Lista completa de colores del W3C: <https://www.w3.org/TR/css-color-3/#html4>

El módulo de caja flexible o flexbox:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout

W3C parámetro style: <https://www.w3.org/TR/css-display-3/#the-display-properties>

MDN (Mozilla Development Network):

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

Para profundizar en los atributos relativos al fondo, los bordes y los contornos puedes hacerlo en los siguientes enlaces:

https://developer.mozilla.org.es/docs/Web/CSS/CSS_Background_and_Borders

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