UT3. Introducción a CSS3

Contenido

1. ¿Qué es CSS?

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, **mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes**.

Al crear una página web, se utiliza en primer lugar el **lenguaje HTML/XHTML para marcar los contenidos**, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje **CSS para definir el aspecto de cada elemento**: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

2. Funcionamiento básico de CSS

Antes de la adopción de CSS, los diseñadores de páginas web debían definir el aspecto de cada elemento dentro de las etiquetas HTML de la página. El siguiente ejemplo muestra una página HTML con estilos definidos sin utilizar CSS:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo de estilos sin CSS</title>
</head>
<body>
    <h1><font color="red" face="Arial" size="5">Titular de la
página</font></h1>
    <p><font color="gray" face="Verdana" size="2">Un párrafo de texto no muy
largo.</font></p>
</body>
</html>
```

Resultado:

El ejemplo anterior utiliza la etiqueta <font> con sus atributos color, face y size para definir el color, la tipografía y el tamaño del texto de cada elemento del documento.

El principal problema de esta forma de definir el aspecto de los elementos se puede ver claramente con el siguiente ejemplo: **si la página tuviera 50 elementos diferentes, habría que insertar 50 etiquetas <font>**. Si el sitio web entero se compone de 10.000 páginas diferentes, habría que definir 500.000 etiquetas <font>. Como cada etiqueta <font> tiene 3 atributos, habría que definir 1.5 millones de atributos.

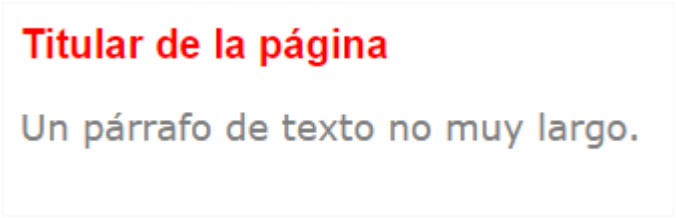
Por otra parte, el diseño de los sitios web está en constante evolución y es habitual modificar cada cierto tiempo los colores principales de las páginas o la tipografía utilizada para el texto. Si se emplea la etiqueta <font>, habría que modificar el valor de 1.5 millones de atributos para modificar el diseño general del sitio web.

La solución que propone CSS es mucho mejor, como se puede ver en el siguiente ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo de estilos con CSS</title>

    <style type="text/css">
        h1 { color: red; font-family: Arial; font-size: large; }
        p { color: gray; font-family: Verdana; font-size: medium; }
    </style>
</head>
<body>
    <h1>Titular de la página</h1>
    <p>Un párrafo de texto no muy largo.</p>
</body>
</html>
```

Resultado:



CSS permite separar los contenidos de la página y su aspecto o presentación. En el ejemplo anterior, dentro de la propia página HTML se reserva una zona en la que se incluye toda la información relacionada con los estilos de la página.

Utilizando CSS, en esa zona reservada se indica que todas las etiquetas <h1> de la página se deben ver de color rojo, con un tipo de letra Arial y con un tamaño de letra grande. Además, las etiquetas <p> de la página se deben ser de color gris, con un tipo de letra Verdana y con un tamaño de letra medio.

Definiendo los estilos de esta forma, no importa el número de elementos <p> que existan en la página, **ya que todos tendrán el mismo aspecto establecido mediante CSS**. Como se verá a continuación, **esta forma de trabajar con CSS no es ideal, ya que, si el sitio web dispone de 10.000 páginas, habría que definir**

10.000 veces el mismo estilo CSS.

3. Cómo incluir CSS en un documento HTML

Una de las principales características de CSS es su flexibilidad y las diferentes opciones que ofrece para realizar una misma tarea. De hecho, **existen tres opciones para incluir CSS en un documento HTML**.

3.1 Incluir CSS en el mismo documento HTML (CSS interno)

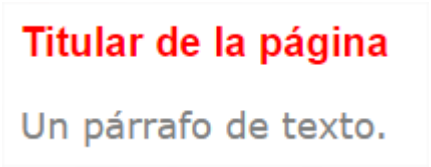
Los estilos se definen en una zona específica del propio documento HTML. **Se emplea la etiqueta <style> de HTML y solamente se pueden incluir en la cabecera del documento** (sólo dentro de la sección **<head>**).

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo de estilos con CSS interno</title>

    <style type="text/css">
        h1 { color: red; font-family: Arial; font-size: large; }
        p { color: gray; font-family: Verdana; font-size: medium; }
    </style>
</head>
<body>
    <h1>Titular de la página</h1>
    <p>Un párrafo de texto.</p>
</body>
</html>
```

Resultado:



Este método se emplea cuando se define un número pequeño de estilos o cuando se quieren incluir estilos específicos en una determinada página HTML que completen los estilos que se incluyen por defecto en todas las páginas del sitio web.

El principal inconveniente es que, si se quiere hacer una modificación en los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que se va a modificar.

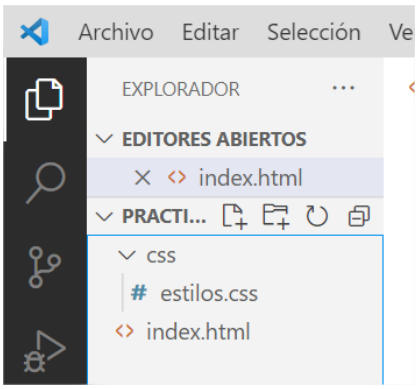
3.2 Definir CSS en un archivo externo (CSS externo)

En este caso, **todos los estilos CSS se incluyen en un archivo de tipo CSS que las páginas HTML enlazan mediante la etiqueta <link>.** Un **archivo de tipo CSS**

no es más que un archivo simple de texto cuya extensión es .css Se pueden crear todos los archivos CSS que sean necesarios y cada página HTML puede enlazar tantos archivos CSS como necesite.

Si se quieren incluir los estilos del ejemplo anterior en un archivo CSS externo, se deben seguir los siguientes pasos (Visual Studio Code):

1) Se crea un archivo con la extensión .css



2) Se añade solamente el siguiente contenido:

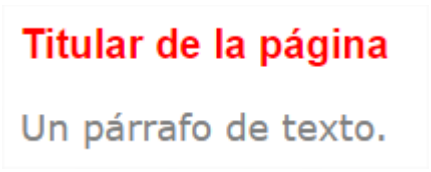
```
h1 { color: red; font-family: Arial; font-size: large; }
p { color: gray; font-family: Verdana; font-size: medium; }
```

3) En la página HTML se enlaza el archivo CSS externo mediante la etiqueta <link>:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo de estilos con CSS externo</title>

    <link rel="stylesheet" type="text/css" href="/css/estilos.css">
</head>
<body>
    <h1>Titular de la página</h1>
    <p>Un párrafo de texto.</p>
</body>
</html>
```

Resultado:



Nota: Es recomendado almacenar las hojas de estilos en un directorio común, por ejemplo, llamado CSS. Además, se recomienda emplear rutas relativas.

Cuando el navegador carga la página HTML anterior, 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 se enlaza un**

archivo CSS:

- **rel**: indica el tipo de relación que tiene 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.

De todas las formas de incluir CSS en las páginas HTML, **esta es la más utilizada con mucha diferencia**. **La principal ventaja es que se puede incluir un mismo archivo CSS en multitud de páginas HTML**, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

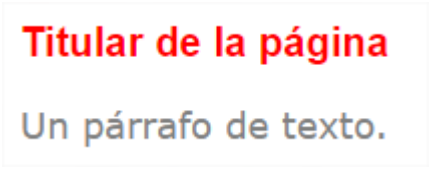
Con este método, **el mantenimiento del sitio web se simplifica al máximo, ya que un solo cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML que enlazan ese archivo**.

3.3 Incluir CSS en los elementos HTML (CSS en línea)

El último método para incluir estilos CSS en documentos HTML **es el peor y el menos utilizado**, ya que tiene los mismos problemas que la utilización de las etiquetas <font>.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Ejemplo de estilos con CSS en linea</title>
</head>
<body>
    <h1 style="color: red; font-family: Arial; font-size: large;">Titular de
la página</h1>
    <p style="color: gray; font-family: Verdana; font-size: medium;">Un
párrafo de texto.</p>
</body>
</html>
```

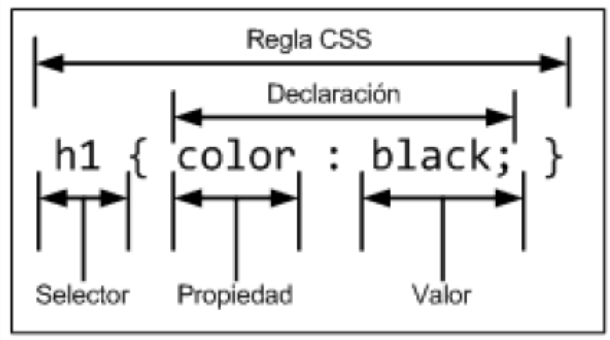
Resultado:



Esta forma de incluir CSS directamente en los elementos HTML solamente se utiliza en determinadas situaciones en las que se debe incluir un estilo muy específico para un solo elemento concreto. **Se emplea el atributo HTML style para definir las propiedades CSS**.

4. Glosario básico

CSS define una serie de términos que permiten describir cada una de las partes que componen los estilos CSS. El siguiente esquema muestra las partes que forman un estilo CSS muy básico:



Los diferentes términos se definen a continuación:

- **Regla**: **cada uno de los estilos que componen una hoja de estilos CSS**. Cada regla está compuesta de una parte de "selectores", un símbolo de "llave de apertura" ({), otra parte denominada "declaración" y, por último, un símbolo de "llave de cierre" (}).
- **Selector**: indica el elemento o elementos HTML a los que se aplica la regla CSS.
- **Declaración**: especifica los estilos que se aplican a los elementos. Está compuesta por una o más propiedades CSS.
- **Propiedad**: permite modificar el aspecto de una característica del elemento.
- **Valor**: indica el nuevo valor de la característica modificada en el elemento.

Un archivo CSS puede contener infinitas reglas CSS, cada regla puede contener infinitos selectores y cada declaración puede estar formada por un número infinito de pares propiedad/valor.

5. Comentarios

CSS permite incluir comentarios entre sus reglas y estilos. Los comentarios son contenidos de texto que el diseñador incluye en el archivo CSS para su propia información y utilidad. Los navegadores ignoran por completo cualquier comentario de los archivos CSS, por lo que es común utilizarlos para estructurar de forma clara los archivos CSS complejos.

El comienzo de un comentario se indica mediante los caracteres /* y el final del comentario se indica mediante */, tal y como se muestra en el siguiente ejemplo:

```
/* Este es un comentario en CSS */
```

Los comentarios pueden ocupar tantas líneas como sea necesario, pero no se puede incluir un comentario dentro de otro comentario:

```
/* Este es un
```

```
comentario CSS de varias
líneas */
```

Aunque los navegadores ignoran los comentarios, su contenido se envía junto con el resto de estilos, por lo que **no se debe incluir en ellos ninguna información sensible o confidencial**.

La sintaxis de los comentarios CSS es muy diferente a la de los comentarios HTML, por lo que no deben confundirse:

```
<!-- Este es un comentario en HTML -->

<!-- Este es un
comentario HTML de varias
líneas -->
```

6. Selectores

Una regla de CSS está formada por una parte llamada "selector" y otra parte llamada "declaración".

La declaración indica "qué hay que hacer" **y el selector indica "a quién hay que hacérselo"**. Por lo tanto, los selectores son imprescindibles para aplicar de forma correcta los estilos CSS en una página.

El estándar de **CSS incluye muchos tipos diferentes de selectores**, que permiten seleccionar de forma muy precisa elementos individuales o conjuntos de elementos dentro de una página web.

No obstante, **la mayoría de páginas de los sitios web se pueden diseñar utilizando solamente los cinco selectores básicos**.

7. Selectores básicos

7.1 Selector universal

Se utiliza para seleccionar todos los elementos de la página. El siguiente ejemplo elimina el margen y el relleno de todos los elementos HTML (por ahora no es importante fijarse en la parte de la declaración de la regla CSS):

```
* {
    margin: 0;
    padding: 0;
}
```

El selector universal se indica mediante un asterisco (*). A pesar de su sencillez, no se utiliza habitualmente, ya que es difícil que un mismo estilo se pueda aplicar a todos los elementos de una página.

7.2 Selector de tipo o etiqueta

Selecciona todos los elementos de la página cuya etiqueta HTML coincide con el valor del selector. El siguiente ejemplo selecciona todos los párrafos de la página:

```
p {
...
}
```

Para utilizar este selector, solamente es necesario indicar el nombre de una etiqueta HTML (sin los caracteres < y >) correspondiente a los elementos que se quieren seleccionar.

El siguiente ejemplo aplica diferentes estilos a los titulares y a los párrafos de una página HTML:

```
h1 {
    color: red;
}

h2 {
    color: blue;
}

p {
    color: black;
}
```

Si se quiere aplicar los mismos estilos a dos etiquetas diferentes, se pueden encadenar los selectores. En el siguiente ejemplo, los títulos de sección h1, h2 y h3 comparten los mismos estilos:

```
h1 {
    color: blue;
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
}

h2 {
    color: blue;
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
}

h3 {
    color: blue;
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
}
```

En este caso, CSS permite **agrupar todas las reglas individuales en una sola regla con un selector múltiple**. Para ello, se incluyen todos **los selectores separados por una coma (,)** y el resultado es que la siguiente regla CSS es equivalente a las tres reglas anteriores:

```
h1, h2, h3 {
    color: blue;
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
```

```
}
```

En las hojas de estilo complejas, **es habitual agrupar las propiedades comunes de varios elementos en una única regla CSS y posteriormente definir las propiedades específicas de esos mismos elementos**. El siguiente ejemplo establece en primer lugar las propiedades comunes de los títulos de sección (color y tipo de letra) y a continuación, establece el tamaño de letra de cada uno de ellos:

```
h1, h2, h3 {
    color: blue;
    font-weight: normal;
    font-family: Arial, Helvetica, sans-serif;
}

h1 {font-size: large;}
h2 {font-size: medium;}
h3 {font-size: small;}
```

Resultado:

7.3 Selector descendente

Selecciona los elementos que se encuentran dentro de otros elementos. Un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento.

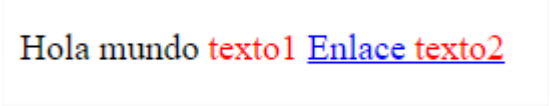
El selector del siguiente ejemplo selecciona todos los elementos de la página que se encuentren dentro de un elemento <p>:

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

Si el código HTML de la página es el siguiente:

```
<p>Hola mundo <span>texto1</span>
    <a href="#">Enlace <span>texto2</span></a>
</p>
```

Resultado:



El selector **p span** selecciona tanto texto1 como texto2. El motivo es que, en el selector descendente, un elemento no tiene que ser "hijo directo" de otro. **La única condición es que un elemento debe estar dentro de otro elemento, sin importar lo profundo que se encuentre.**

Al resto de elementos de la página que no están dentro de un elemento

<p>, no se les aplica la regla CSS anterior.

Los selectores descendentes permiten mejorar la precisión del selector de tipo o etiqueta. Así, utilizando el selector descendente es posible aplicar diferentes estilos a los elementos del mismo tipo. El siguiente ejemplo amplía el anterior y muestra de color azul todo el texto de los contenidos dentro de un <h1>:

```
p span { color: red; }
h1 span { color: blue; }
```

Con las reglas CSS anteriores:

- Los elementos que se encuentran dentro de un elemento <p> se muestran de color rojo.
- Los elementos que se encuentran dentro de un elemento <h1> se muestran de color azul.
- El resto de elementos de la página, se muestran con el color por defecto aplicado por el navegador.

La sintaxis formal del selector descendente se muestra a continuación:

```
elemento1 elemento2 elemento3 ... elementoN
```

Los selectores descendentes siempre están formados por dos o más partes separadas entre sí por espacios en blanco. La última parte indica el elemento sobre el que se aplican los estilos y todas las partes anteriores indican el lugar en el que se tiene que encontrar ese elemento para que los estilos se apliquen realmente.

En el siguiente ejemplo, el selector descendente se compone de cuatro partes:

```
p a span em { color:red;}
```

Los estilos de la regla anterior se aplican a los elementos de tipo que se encuentren dentro de elementos de tipo , que a su vez se encuentren dentro de elementos de tipo <a> que se encuentren dentro de elementos de tipo <p>.

No debe confundirse el selector descendente con la combinación de selectores:

```
/* El estilo se aplica a todos los elementos "p", "a", "span" y "em" */
p, a, span, em { text-decoration: underline; }

/* El estilo se aplica solo a los elementos "em" que se
encuentran dentro de "p a span" */
p a span em { text-decoration: underline; }
```

Si se emplea el selector descendente combinado con el selector universal, se puede restringir el alcance de un selector descendente. El siguiente ejemplo, muestra los dos enlaces de color rojo:

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

```
<p><a href="#">Enlace1</a></p>
<p><span><a href="#">Enlace2</a></span></p>
```

Resultado:

Sin embargo, en el siguiente ejemplo solamente el segundo enlace se muestra de color rojo:

```
p * a { color: red; }
```

```
<p><a href="#">Enlace1</a></p>
<p><span><a href="#">Enlace2</a></span></p>
```

Resultado:

La razón es que el selector `p * a` se traduce como todos los elementos de tipo `<a>` que se encuentren dentro de cualquier elemento que, a su vez, se encuentre dentro de un elemento de tipo `<p>`. Como el primer elemento `<a>` se encuentra directamente bajo un elemento `<p>`, no se cumple la condición del selector `p * a.`

7.4 Selector de clase

Si se considera el siguiente código HTML de ejemplo:

```
<body>
<p>Esto es el primer párrafo.</p>
<p>Esto es el segundo párrafo.</p>
<p>Esto es el tercer párrafo.</p>
</body>
```

¿Cómo se pueden aplicar estilos CSS sólo al primer párrafo? El selector universal (*) no se puede utilizar porque selecciona todos los elementos de la página. El selector de tipo o etiqueta (p) tampoco se puede utilizar porque seleccionaría todos los párrafos. Por último, el selector descendente (body p) tampoco se puede utilizar porque todos los párrafos se encuentran en el mismo sitio.

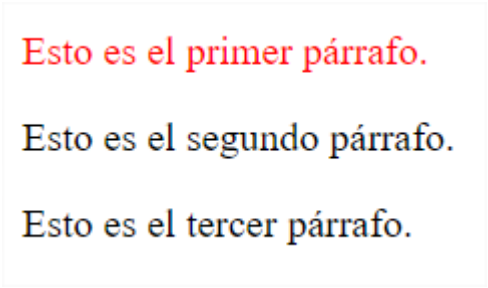
Una de las soluciones más sencillas para aplicar estilos a un solo elemento de la página consiste en **utilizar el atributo class** de HTML sobre ese elemento para indicar directamente la regla CSS que se le debe aplicar:

```
<body>
<p class="destacado">Esto es el primer párrafo.</p>
<p>Esto es el segundo párrafo.</p>
<p>Esto es el tercer párrafo.</p>
</body>
```

A continuación, se crea en el archivo CSS una nueva regla llamada destacado con todos los estilos que se van a aplicar al elemento. Para que el navegador no confunda este selector con los otros tipos de selectores, **se prefija el valor del atributo class con un punto (.)** tal y como muestra el siguiente ejemplo:

```
.destacado { color: red; }
```

Resultado:

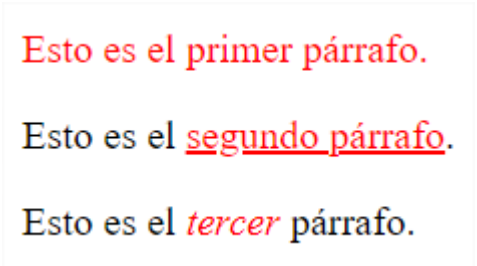


El selector .destacado se interpreta como "cualquier elemento de la página cuyo atributo class sea igual a destacado", por lo que solamente el primer párrafo cumple esa condición.

Este tipo de selectores se llaman selectores de clase y son los más utilizados junto con los selectores de ID que se verán a continuación. La principal característica de este selector es que en una misma página HTML varios elementos diferentes pueden utilizar el mismo valor en el atributo class:

```
<body>
<p class="destacado">Esto es el primer párrafo.</p>
<p>Esto es el <a href="#" class="destacado">segundo párrafo</a>.</p>
<p>Esto es el <em class="destacado">tercer</em> párrafo.</p>
</body>
```

Resultado:



Los selectores de clase son imprescindibles para diseñar páginas web complejas, ya que permiten disponer de una precisión total al seleccionar los elementos. Además, estos selectores permiten reutilizar los mismos estilos

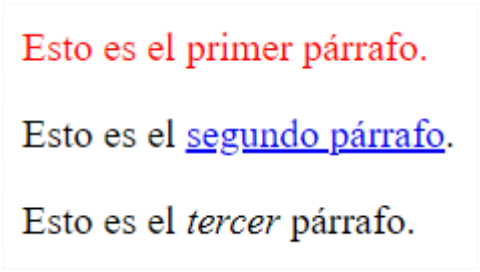
para varios elementos diferentes.

En ocasiones, es necesario restringir el alcance del selector de clase. Si se considera de nuevo el ejemplo anterior. **¿Cómo es posible aplicar estilos solamente al párrafo cuyo atributo class sea igual a destacado?**

Combinando el selector de tipo y el selector de clase, se obtiene un selector mucho más específico:

```
p.destacado { color: red }
```

Resultado:



El selector p.destacado se interpreta como "aquellos elementos de tipo <p> que dispongan de un atributo class con valor destacado". De la misma forma, el selector a.destacado solamente selecciona los enlaces cuyo atributo class sea igual a destacado.

De lo anterior se deduce que el atributo .destacado es equivalente a *.destacado, por lo que todos los diseñadores obvian el símbolo * al escribir un selector de clase normal.

No debe confundirse el selector de clase con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo class="aviso" */
p.aviso { ... }
/* Todos los elementos con atributo class="aviso" que estén dentro
de cualquier elemento de tipo "p" */
p .aviso { ... }
/* Todos los elementos "p" de la página y todos los elementos con
atributo class="aviso" de la página */
p, .aviso { ... }
```

Por último, **es posible aplicar los estilos de varias clases CSS sobre un mismo elemento**. La sintaxis es similar, pero **los diferentes valores del atributo class se separan con espacios en blanco**. En el siguiente ejemplo:

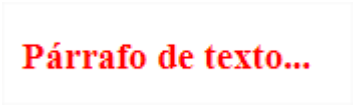
```
<p class="especial destacado error">Párrafo de texto...</p>
```

Al párrafo anterior se le aplican los estilos definidos en las reglas .especial, .destacado y .error, por lo que en el siguiente ejemplo, el texto del párrafo se vería de color rojo, en negrita y con un tamaño de letra de 15 píxel:

```
.error { color: red; }
.destacado { font-size: 15px; }
```

```
.especial { font-weight: bold; }
```

Resultado:



Si un elemento dispone de un atributo class con más de un valor, es posible utilizar un selector más avanzado:

```
.error { color: red; }
.error.destacado { color: blue; }
.destacado { font-size: 15px; }
.especial { font-weight: bold; }
```

Resultado:

En el ejemplo anterior, el color de la letra del texto es azul y no rojo. El motivo es que se ha utilizado un selector de clase múltiple *.error.destacado*, que se interpreta como "aquellos elementos de la página que dispongan de un atributo class con al menos los valores error y destacado". **El estilo que se aplica es el del selector más específico.**

7.5 Selectores de ID

En ocasiones, es necesario aplicar estilos CSS a un único elemento de la página. Aunque puede utilizarse un selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso.

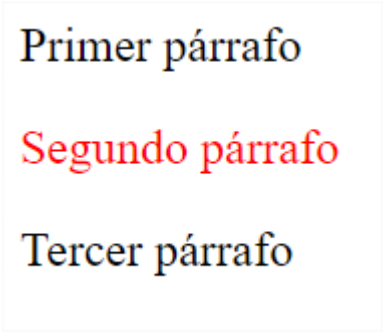
El selector de ID permite seleccionar un elemento de la página a través del valor de su atributo id. Este tipo de selectores sólo seleccionan un elemento de la página porque el valor del **atributo id no se puede repetir en dos elementos diferentes de una misma página**.

La sintaxis de los selectores de ID es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de la almohadilla (#) en vez del punto (.) como prefijo del nombre de la regla CSS. Por ejemplo:

```
    <p>Primer párrafo</p>
    <p id="destacado">Segundo párrafo</p>
    <p>Tercer párrafo</p>
```

```
#destacado { color: red; }
```

Resultado:



En el ejemplo anterior, el selector #destacado solamente selecciona el segundo párrafo (cuyo atributo id es igual a destacado).

La principal diferencia entre este tipo de selector y el selector de clase tiene que ver con HTML y no con CSS. Como se sabe, en una misma página, el valor del atributo id debe ser único, de forma que dos elementos diferentes no pueden tener el mismo valor de id. Sin embargo, el atributo class no es obligatorio que sea único, de forma que muchos elementos HTML diferentes pueden compartir el mismo valor para su atributo class.

De esta forma, **la recomendación general es la de utilizar el selector de ID cuando se quiere aplicar un estilo a un solo elemento específico de la página y utilizar el selector de clase cuando se quiere aplicar un estilo a varios elementos diferentes de la página HTML**.

Al igual que los selectores de clase, en este caso también **se puede restringir el alcance del selector mediante la combinación con otros selectores**. El siguiente ejemplo aplica la regla CSS solamente al elemento de tipo <p> que tenga un atributo id igual al indicado:

```
p#aviso { color: blue; }
```

A primera vista, restringir el alcance de un selector de ID puede parecer absurdo. En realidad, un selector de tipo p#aviso **sólo tiene sentido cuando el archivo CSS se aplica sobre muchas páginas HTML diferentes.**

En este caso, algunas páginas pueden disponer de elementos con un atributo id igual a aviso y que no sean párrafos, por lo que la regla anterior no se aplica sobre esos elementos.

No debe confundirse el selector de ID con los selectores anteriores:

```
/* Todos los elementos de tipo "p" con atributo id="aviso" */
p#aviso { ... }
/* Todos los elementos con atributo id="aviso" que estén dentro
de cualquier elemento de tipo "p" */
p #aviso { ... }
/* Todos los elementos "p" de la página y todos los elementos con
atributo id="aviso" de la página */
p, #aviso { ... }
```

7.6 Combinación de selectores básicos

CSS permite la combinación de uno o más tipos de selectores para restringir el

alcance de las reglas CSS. A continuación, se muestran algunos ejemplos habituales de combinación de selectores.

```
.aviso .especial { ... }
```

El anterior selector solamente selecciona aquellos elementos con un class="especial" que se encuentren dentro de cualquier elemento con un class="aviso".

Si se modifica el anterior selector:

```
div.aviso span.especial { ... }
```

Ahora, el selector solamente selecciona aquellos elementos de tipo con un atributo class="especial" que estén dentro de cualquier elemento de tipo que tenga un atributo class="aviso".

La combinación de selectores puede llegar a ser todo lo compleja que sea necesario:

```
ul#menuPrincipal li.destacado a#inicio { ... }
```

El anterior selector hace referencia al enlace con un atributo id igual a inicio que se encuentra dentro de un elemento de tipo <li> con un atributo class igual a destacado, que forma parte de una lista <ul> con un atributo id igual a menuPrincipal.

8. Selectores avanzados

Utilizando solamente los selectores básicos de la sección anterior, es posible diseñar

prácticamente cualquier página web. No obstante, CSS define otros selectores más avanzados que permiten simplificar las hojas de estilos.

8.1 Selector de hijos

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento.

Se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el "signo de mayor que" (>):

```
p > span { color: blue; }
```

```
<p><span>Texto1</span></p>
<p><a href="#"><span>Texto2</span></a></p>
```

Resultado:

En el ejemplo anterior, el selector p > span se interpreta como "*cualquier elemento que sea hijo directo de un elemento <p>*", por lo que el primer elemento cumple la condición del selector. Sin embargo, el segundo elemento no la cumple porque es descendiente pero no es hijo directo de un elemento <p>.

El siguiente ejemplo muestra las diferencias entre el selector descendente y el selector de hijos:

```
p a { color: red; }
p > a { color: red;  font-weight: bold;}
```

```
<p><a href="#">Enlace1</a></p>
<p><span><a href="#">Enlace2</a></span></p>
```

Resultado:

El primer selector es de tipo descendente y por tanto se aplica a todos los elementos <a> que se encuentran dentro de elementos <p>. En este caso, los estilos de este selector se aplican a los dos enlaces.

Por otra parte, el selector de hijos obliga a que el elemento <a> sea hijo directo de un elemento <p>. Por lo tanto, los estilos del selector p > a no se aplican al segundo enlace del ejemplo anterior.

8.2 Selector adyacente

El selector adyacente utiliza el signo + y su sintaxis es:

```
elemento1 + elemento2 { ... }
```

La explicación del comportamiento de este selector no es sencilla, ya que selecciona todos los elementos de tipo elemento2 que cumplan las dos siguientes condiciones:

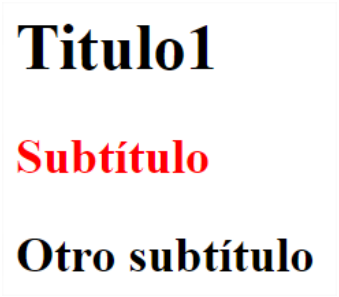
- elemento1 y elemento2 deben ser hermanos, por lo que su elemento padre debe ser el mismo.
- elemento2 debe aparecer inmediatamente después de elemento1 en el código HTML de la página.

En el siguiente ejemplo:

```
h1 + h2 { color: red }
```

```
<body>
    <h1>Titulo1</h1>
    <h2>Subtítulo</h2>
    <h2>Otro subtítulo</h2>
</body>
```

Resultado:



Los estilos del selector h1 + h2 se aplican al primer elemento <h2> de la página, pero no al segundo <h2>, ya que:

- El elemento padre de <h1> es <body>, el mismo padre que el de los dos elementos <h2>. Así, los dos elementos <h2> cumplen la primera condición del selector adyacente.
- El primer elemento <h2> aparece en el código HTML justo después del elemento <h1>, por lo que este elemento <h2> también cumple la segunda condición del selector adyacente.
- Por el contrario, el segundo elemento <h2> no aparece justo después del elemento <h1>, por lo que no cumple la segunda condición del selector adyacente y por tanto no se le aplican los estilos de h1 + h2.

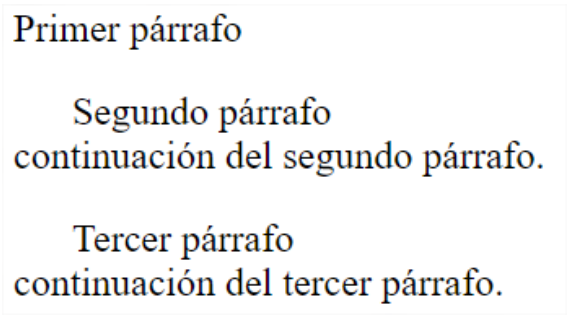
El siguiente ejemplo es muy útil para los textos que se muestran como libros:

```
p + p { text-indent: 1.5em; }
```

En muchos libros, suele ser habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. Con el selector p + p, se seleccionan todos los párrafos que estén dentro del mismo elemento padre que otros párrafos y que vayan justo después de otro párrafo. En otras palabras, **el selector p + p selecciona todos los párrafos de un elemento salvo el primer párrafo**.

```
<body>
    <p>Primer párrafo</p>
    <p>Segundo párrafo<br>continuación del segundo párrafo.</p>
    <p>Tercer párrafo<br>continuación del tercer párrafo.</p>
</body>
```

Resultado:



8.3 Selector de atributos

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- [nombre_atributo], selecciona los elementos que tienen establecido el atributo llamado nombre_atributo, independientemente de su valor.
- [nombre_atributo=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor igual a valor.
- [nombre_atributo~=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y al menos uno de los valores del atributo es valor.
- [nombre_atributo|=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y cuyo valor es una serie de palabras separadas con guiones, pero que comienza con valor. Este tipo de selector sólo es útil para los atributos de tipo lang que indican el idioma del contenido del elemento.
- nombre_atributo*=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y el valor del atributo contenga valor.
- nombre_atributo^=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y el valor de atributo comienza por valor.
- nombre_atributo$=valor], selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y el valor de atributo termina por valor.

A continuación, se muestran algunos ejemplos de estos tipos de selectores:

```
/* Se muestran de color azul todos los enlaces que tengan
un atributo "class", independientemente de su valor */
a[class] { color: blue; }

/* Se muestran de color azul todos los enlaces que tengan
un atributo "class" con el valor "externo" */
a[class="externo"] { color: blue; }

/* Se muestran de color azul todos los enlaces que apunten
al sitio "http://www.ejemplo.com" */
a[href="http://www.ejemplo.com"] { color: blue; }
```

```
/* Se muestran de color azul todos los enlaces que tengan
un atributo "class" en el que al menos uno de sus valores
sea "externo" */
a[class~="externo"] { color: blue; }

/* Selecciona todos los elementos de la página cuyo atributo
"lang" sea igual a "en", es decir, todos los elementos en inglés */
*[lang=en] { ... }

/* Selecciona todos los elementos de la página cuyo atributo
"lang" empiece por "es", es decir, "es", "es-ES", "es-AR", etc. */
*[lang|="es"] { color : red }
```

9. Agrupación de reglas

Cuando se crean archivos CSS complejos con decenas o cientos de reglas, es habitual que los estilos que se aplican a un mismo selector se definan en diferentes reglas:

```
h1 { color: red; }
...
h1 { font-size: 2em; }
...
h1 { font-family: Verdana; }
```

Las tres reglas anteriores establecen el valor de tres propiedades diferentes de los elementos <h1>. Antes de que el navegador muestre la página, procesa todas las reglas CSS de la página para tener en cuenta todos los estilos definidos para cada elemento.

Cuando el selector de dos o más reglas CSS es idéntico, se pueden agrupar las declaraciones de las reglas para hacer las hojas de estilos más eficientes:

```
h1 {
    color: red;
    font-size: 2em;
    font-family: Verdana;
    }
```

El ejemplo anterior tiene el mismo efecto que las tres reglas anteriores, pero es más eficiente y es más fácil de modificar y mantener por parte de los diseñadores. Como **CSS ignora los espacios en blanco y las nuevas líneas**, también se pueden agrupar las reglas de la siguiente forma:

```
h1 { color: red; font-size: 2em; font-family: Verdana; }
```

Si se quiere reducir al máximo el tamaño del archivo CSS para mejorar ligeramente el tiempo de carga de la página web, también es posible indicar la regla anterior de la siguiente forma:

```
h1 {color:red;font-size:2em;font-family:Verdana;}
```

Ha este proceso de eliminar espacios y saltos de línea para mejorar el tiempo de carga se le llama minificar.

10. Herencia

Uno de los conceptos más característicos de CSS es la herencia de los estilos definidos para los elementos. Cuando se establece el valor de alguna propiedad en un elemento, todos sus descendientes heredan inicialmente ese mismo valor.

Si se indica por ejemplo un tipo de letra al elemento <body> de una página, todos los elementos de la página mostrarán ese tipo de letra, salvo que se indique lo contrario:

```
body { font-family: Arial; color: black; }
h1 { font-family: Verdana; }
p { color: red; }
```

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

Resultado:

En el ejemplo anterior, se ha indicado que la etiqueta <body> tiene asignado un tipo de letra Arial y un color de letra negro. Así, todos los elementos de la página (salvo que se indique lo contrario) se muestran de color negro y con la fuente Arial.

La segunda regla indica que los elementos <h1> se muestran con otra tipografía diferente a la heredada. La tercera regla indica que los elementos <p> varían su color respecto del color que han heredado.

La herencia de estilos no funciona en todas las propiedades CSS, por lo que se debe estudiar cada propiedad de forma individual.

11. Colisiones de estilos

En las hojas de estilos complejas, es habitual que varias reglas CSS se apliquen a un mismo elemento HTML. El problema de estas reglas múltiples es que se pueden dar colisiones como la del siguiente ejemplo:

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

```
<p>Párrafo 1.</p>
```

¿De qué color se muestra el párrafo anterior? CSS tiene un mecanismo de resolución de colisiones muy complejo y que tiene en cuenta el tipo de hoja de estilo que se trate (de navegador, de usuario o de diseñador), la importancia de cada regla y lo específico que sea el selector.

Resultado:

Aunque los tipos de hojas de estilos y su importancia se verán más adelante, se describe a continuación el **método genérico seguido por CSS para resolver las colisiones**:

1. Determinar todas las declaraciones que se aplican al elemento para el medio CSS seleccionado.
2. Ordenar las declaraciones según su origen (CSS de navegador, de usuario o de diseñador) y su importancia (palabra clave !important).
3. Ordenar las declaraciones según lo específico que sea el selector. Cuanto más genérico es un selector, menos importancia tienen sus declaraciones.
4. Si después de aplicar las normas anteriores existen dos o más reglas con la misma prioridad, **se aplica la que se indicó en último lugar**.

Cuando se estudie cada uno de los conceptos del método anterior, se comprenderá completamente su funcionamiento. De momento, la norma que se puede seguir es la de la "especificidad" del selector:

1. **Cuanto más específico sea un selector, más importancia tiene su regla asociada.**
2. **A igual especificidad, se considera la última regla indicada.**

Como en el ejemplo anterior los dos selectores son idénticos, las dos reglas tienen la misma prioridad y prevalece la que se indicó en último lugar, por lo que el párrafo se muestra de color azul.

En el siguiente ejemplo, la regla CSS que prevalece se decide por lo específico que es cada selector:

```
p { color: red; }
p#especial { color: green; }
* { color: blue; }
```

```
<p id="especial">Párrafo 1.</p>
```

Al elemento <p> se le aplican las tres declaraciones. Como su origen y su importancia es la misma, decide la especificidad del selector. El selector * es el menos específico, ya que se refiere a "todos los elementos de la página". El selector p es poco específico porque se refiere a "todos los párrafos de la página".

Por último, el selector p#especial sólo hace referencia a "el párrafo de la página

cuyo atributo id sea igual a especial". Como el selector p#especial es el más específico, su declaración es la que se tiene en cuenta y por tanto el párrafo se muestra de color verde.

12. Unidades de medida y colores

Muchas de las propiedades de CSS que se ven en los próximos capítulos permiten indicar unidades de medida y colores en sus valores. Además, **CSS es tan flexible que permite indicar las medidas y colores de muchas formas diferentes**. Por este motivo, se presentan a continuación todas las alternativas disponibles en CSS para indicar las medidas y los colores. En los siguientes capítulos, simplemente se indicará que el valor de una propiedad puede tomar el valor de una medida o de un color, sin detallar las diferentes alternativas disponibles para cada valor.

12.1 Unidades de medida

Las medidas en CSS se emplean, entre otras, para definir la altura, anchura y márgenes de los elementos y para establecer el tamaño de letra del texto. **Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).**

CSS divide todas las unidades de medida en dos grupos: absolutas y relativas. Las medidas relativas definen su valor en relación con otra medida, por lo que, para obtener su valor real, se debe realizar alguna operación con el valor indicado. Las unidades absolutas establecen de forma completa el valor de una medida, por lo que su valor real es directamente el valor indicado.

Si el valor es 0, la unidad de medida es opcional. Si el valor es distinto a 0 y no se indica ninguna unidad, la medida se ignora completamente, lo que suele ser una fuente habitual de errores para los diseñadores que empiezan con CSS. Algunas propiedades permiten indicar medidas negativas, aunque habitualmente sus valores son positivos.

12.1.1 Unidades relativas

Las unidades relativas son más flexibles que las unidades absolutas porque se adaptan más fácilmente a los diferentes medios. A continuación, se muestra la lista de unidades de medida relativas y la referencia que se toma para determinar su valor real:

- **em**, (no confundir con la etiqueta de HTML) relativa respecto del tamaño de letra empleado. Aunque no es una definición exacta, el valor de 1em se puede aproximar por la anchura de la letra M ("eme mayúscula") del tipo de letra que se esté utilizando
- ex, relativa respecto de la altura de la letra x ("equis minúscula") del tipo de letra que se
- esté utilizando
- **px**, (píxel) relativa respecto de la pantalla del usuario

Las unidades em y ex no han sido definidas por CSS, sino que llevan décadas utilizándose en el campo de la tipografía. La unidad em hace referencia al tamaño

en puntos de la letra que se está

utilizando. Si se utiliza una tipografía de 12 puntos, 1em equivale a 12 puntos. El valor de 1ex se puede aproximar por 0.5 em.

En el siguiente ejemplo, se indica que el tamaño de letra del texto de la página debe ser el 90% del tamaño por defecto (que depende de cada navegador, aunque es muy similar entre ellos):

```
body { font-size: 0.9em; }
```

Si este tamaño por defecto es 12, el valor 0.9em sería igual a 0.9 x 12 = 10.8.

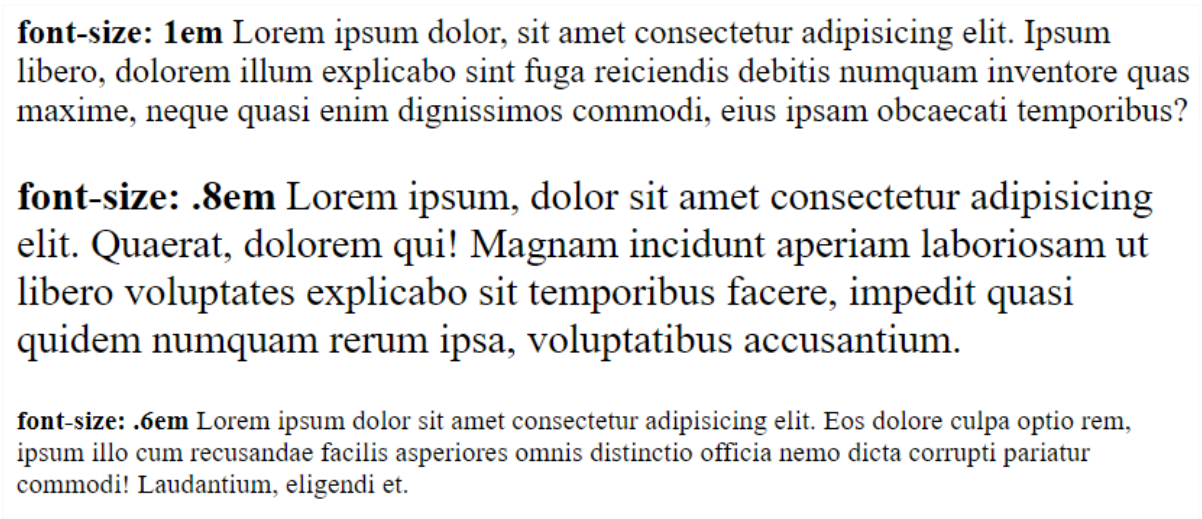
Cuando el valor decimal de una medida es inferior a 1, se puede omitir el 0 de la izquierda, por lo que el código anterior es equivalente al código siguiente:

```
body { font-size: .9em; }
```

El siguiente ejemplo muestra el uso de la unidad em para establecer el tamaño de la letra de diferentes párrafos:

```
.p1{font-size: 1em;}
.p2{font-size: 1.2em;}
.p3{font-size: .8em;}
```

Resultado:



El primer párrafo muestra la letra con un tamaño de 1em, es decir, el tamaño por defecto en el navegador del usuario. El segundo párrafo ha establecido el tamaño de letra en 1.2em, es decir, un 20% más grande que el tamaño por defecto. Por último, el tercer párrafo ha indicado un tamaño de .8em, es decir, un 20% inferior al tamaño por defecto.

Cuando se estudian por primera vez, las unidades relativas parecen demasiado complicadas. Al fin y al cabo, siempre se debe tomar la referencia de la unidad para obtener su valor real. Sin embargo, sus ventajas son mucho mayores que sus inconvenientes.

El ejemplo anterior establece el tamaño de la letra mediante los valores 1em, 1.2em y .8em. En otras palabras, el código anterior está estableciendo los tamaños de letra a "normal", "grande" y "pequeño" respectivamente. Independientemente del tamaño de letra por defecto del dispositivo del usuario, el primer párrafo se verá con un tamaño de letra "normal" (1em), el segundo párrafo se verá más "grande" de lo normal (1.2em) y el último párrafo se verá "pequeño" (.8em).

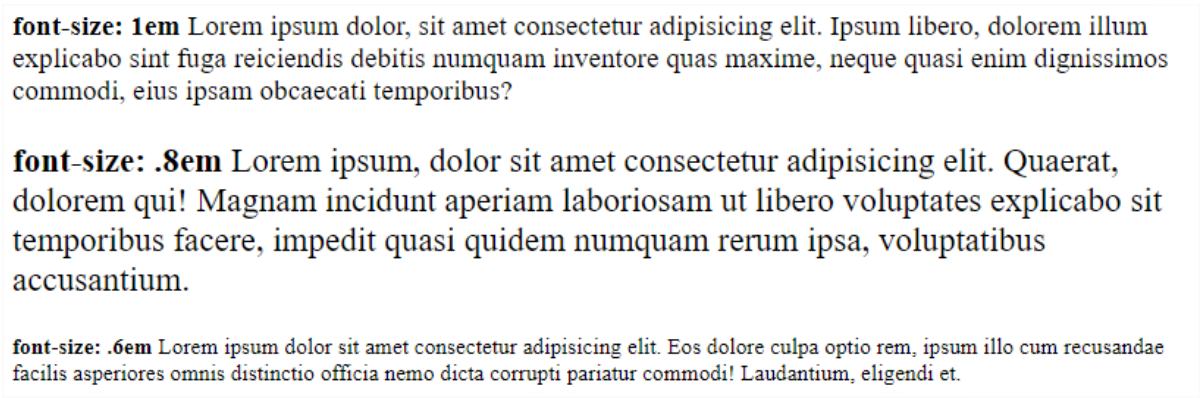
De esta forma, **si el usuario aumenta el tamaño de letra en su navegador, las proporciones se mantendrán**. Si el tamaño de letra por defecto es 12, el primer párrafo se verá con tamaño 12, pero si el usuario aumenta el tamaño de letra por defecto a 20, el primer párrafo se verá con tamaño 20. Como se ve, las unidades relativas permiten mantener las proporciones del diseño independientemente del tamaño de letra por defecto del navegador del usuario.

La referencia para el valor de font-size de un elemento siempre es el tamaño de letra de su elemento padre (es decir, del elemento en el que se encuentra). **Si el elemento no se encuentra dentro de ningún otro elemento, la referencia es el tamaño de letra del elemento <body>. Si no se indica de forma explícita un valor para el tamaño de letra del elemento <body>, la referencia es el tamaño de letra por defecto del navegador**.

Siguiendo esta norma, si en el ejemplo anterior se modifica el tamaño de letra del elemento <body> (que es el elemento padre de los tres párrafos) y se le asigna un valor de 0.8em, el aspecto que muestran los párrafos en el navegador es el siguiente:

```
body{font-size: .8em;}
.p1{font-size: 1em;}
.p2{font-size: 1.2em;}
.p3{font-size: .8em;}
```

Resultado:



Al haber reducido el tamaño de letra que era la referencia del tamaño de letra de los tres párrafos, su texto se ve con una letra más pequeña, aunque manteniendo las proporciones.

El funcionamiento de la unidad ex es idéntico a em, salvo que, en este caso, la referencia es la altura de la letra x minúscula.

Las medidas indicadas en píxel también se consideran relativas, ya que el aspecto de los elementos dependerá de la resolución del dispositivo en el que

se visualiza el documento HTML.

Las distintas unidades se pueden mezclar entre los diferentes elementos de una misma página, como en el siguiente ejemplo:

```
body { font-size: 10px; }
h1 { font-size: 2.5em; }
```

En primer lugar, se establece un tamaño de letra base de 10 píxel para toda la página. A continuación, se asigna un tamaño de 2.5em al elemento <h1>, por lo que su tamaño de letra real será de 2.5 x 10px = 25px.

Las medidas relativas no se heredan directamente, sino que se heredan sus valores reales una vez calculados. El siguiente ejemplo muestra este comportamiento:

```
body {
    font-size: 12px;
    text-indent: 3em;
    }
h1 { font-size: 15px }
```

La propiedad text-indent, como se verá en los próximos capítulos, se utiliza para tabular la primera línea de un texto. El elemento <body> define un valor para esta propiedad, pero el elemento <h1> no lo hace, por lo que heredará el valor de su elemento padre. Sin embargo, el valor heredado no es 3em, sino 36px.

Si se heredará el valor 3em, al multiplicarlo por el valor de font-size del elemento <h1> (que vale 15px) el resultado sería 3em x 15px = 45px. No obstante, como se ha comentado, los valores que se heredan no son los relativos, sino los valores ya calculados.

12.1.2 Unidades absolutas

Las unidades absolutas definen las medidas de forma completa, ya que sus valores reales no se calculan a partir de otro valor de referencia, sino que son directamente los valores indicados. A continuación, se muestra la lista completa de unidades absolutas definidas por CSS y su significado:

- in, del inglés "inches", pulgadas (1 pulgada son 2.54 centímetros)
- cm, centímetros
- mm, milímetros
- pt, puntos (1 punto equivale a 1 pulgada/72, es decir, unos 0.35 milímetros)
- pc, picas (1 pica equivale a 12 puntos, es decir, unos 4.23 milímetros)

De todas las unidades absolutas, la única que se utiliza con cierta frecuencia es la de los puntos (pt). El motivo es que se trata de la unidad preferida para indicar el tamaño de letra del texto para los documentos que se van a imprimir.

12.1.3 Porcentajes

CSS define otra unidad de medida relativa basada en los porcentajes. Un

porcentaje está formado por un **valor numérico seguido del símbolo % y siempre está referenciado a otra medida**. Cada una de las propiedades de CSS que permiten indicar como valor un porcentaje, define el valor al que hace referencia ese porcentaje.

Los porcentajes se pueden utilizar por ejemplo para establecer el valor del tamaño de letra de los elementos:

```
body { font-size: 1em; }
h1 { font-size: 200%; }
h2 { font-size: 150%; }
```

Los tamaños establecidos para los elementos <h1> y <h2> mediante las reglas anteriores, son equivalentes a 2em y 1.5em respectivamente, por lo que es más habitual definirlos mediante em.

Los porcentajes también se utilizan para establecer la anchura de los elementos:

```
div#contenido { width: 600px; }
div.principal { width: 80%; }
```

En el ejemplo anterior, la referencia del valor 80% es la anchura de su elemento padre. Por tanto, el elemento cuyo atributo class vale principal tiene una anchura de 80% x 600px = 480px.

12.1.4 Recomendaciones

En general, **se recomienda el uso de unidades relativas siempre que sea posible, ya que mejora la accesibilidad de la página y permite que los documentos se adapten fácilmente a cualquier medio y dispositivo.**

Normalmente se utilizan píxel y porcentajes para definir el layout del documento (básicamente, la anchura de las columnas y elementos de las páginas) **y em y porcentajes para el tamaño de letra de los textos.**

12.2 Colores

Los colores en CSS se pueden indicar de cinco formas diferentes: **palabras clave, colores del sistema, RGB hexadecimal, RGB numérico y RGB porcentual**. Aunque el método **más habitual es el del RGB hexadecimal**, a continuación, se muestran todas las alternativas que ofrece CSS.

12.2.1 Palabras clave

CSS define 17 palabras clave para referirse a los colores básicos. Las palabras se corresponden con el nombre en inglés de cada color:

aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, yellow

Aunque es una forma muy sencilla de referirse a los colores básicos, este método prácticamente no se utiliza en las hojas de estilos de los sitios web reales, ya que se trata de una gama de colores muy limitada.

12.2.2 RGB decimal

En el modelo RGB un color se define indicando sus tres componentes R (rojo), G (verde) y B (azul). Cada una de las componentes puede tomar un valor entre cero y un valor máximo. De esta forma, el color rojo puro en RGB se crea mediante el máximo valor de la componente R y un valor de 0 para las componentes G y B.

Si todas las componentes valen 0, el color creado es el negro y si todas las componentes toman su valor máximo, el color obtenido es el blanco. En CSS, las componentes de los colores definidos mediante RGB decimal pueden tomar valores entre 0 y 255. El siguiente ejemplo establece el color del texto de un párrafo:

```
p { color: rgb(71, 98, 176); }
```

La sintaxis que se utiliza para indicar los colores es rgb() y entre paréntesis se indican las tres componentes RGB, en ese mismo orden y separadas por comas. El color del ejemplo anterior se obtendría mezclando las componentes R=71, G=98, B=176, que se corresponde con un color azul claro.

Si se indica un valor menor que 0 para una componente, automáticamente se transforma su valor en 0. Igualmente, si se indica un valor mayor que 255, se transforma automáticamente su valor a 255.

12.2.3 RGB porcentual

Otra forma de indicar las componentes RGB de un color es mediante un porcentaje. El funcionamiento y la sintaxis de este método es el mismo que el del RGB decimal. La única diferencia en este caso es que el valor de las componentes RGB puede tomar valores entre 0% y 100%. El mismo color del ejemplo anterior se puede representar de forma porcentual:

```
p { color: rgb(27%, 38%, 69%); }
```

Al igual que sucede con el RGB decimal, si se indica un valor inferior a 0%, se transforma automáticamente en 0% y si se indica un valor superior a 100%, se trunca su valor a 100%.

12.2.4 RGB hexadecimal

Aunque es el método más complicado para indicar los colores, se trata del método más utilizado con mucha diferencia. De hecho, prácticamente todos los sitios web reales utilizan exclusivamente este método.

Para definir un color en CSS con RGB hexadecimal se realizan los siguientes pasos:

1. Se determinan las componentes RGB del color original, por ejemplo: R = 71, G = 98, B =176.
2. El valor numérico de cada componente se transforma al sistema numérico hexadecimal. Se trata de una operación exclusivamente matemática, por lo que puedes utilizar una calculadora. En el ejemplo anterior, el valor hexadecimal de cada componente es: R = 47, G = 62, B = B0.
3. Para obtener el color completo en formato RGB hexadecimal, se concatenan los valores de las componentes RGB en ese orden y se les añade el prefijo #. De esta forma, el color del ejemplo anterior es #4762B0 en formato RGB hexadecimal.

Siguiendo el ejemplo de las secciones anteriores, el color del párrafo se indica de la siguiente forma utilizando el color en formato RGB hexadecimal:

```
p { color: #4762B0; }
```

Una de las ventajas del formato RGB hexadecimal es que se pueden comprimir sus valores cuando todas sus componentes son iguales dos a dos:

#AAA = #AAAAAA
#FFF = #FFFFFF
#A0F = #AA00FF
#369 = #336699
#000 = #000000
#FFF = #FFFFFF

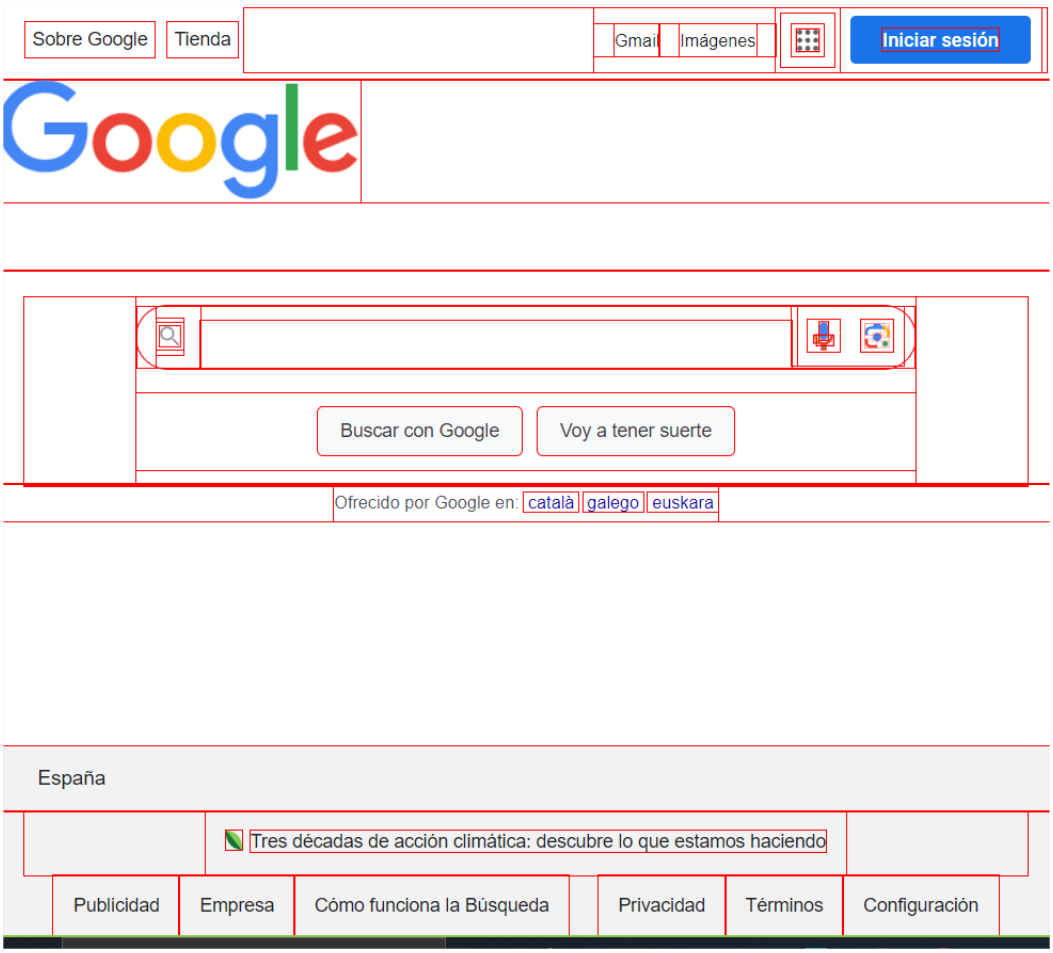
13. Modelo de cajas (box model)

El modelo de cajas o "box model" es seguramente la característica más importante del lenguaje de hojas de estilos CSS, ya que condiciona el diseño de todas las páginas web. **El "box model" es el comportamiento de CSS que hace que todos los elementos incluidos en una página HTML se representen mediante cajas rectangulares. CSS permite controlar el aspecto de todas las cajas**.

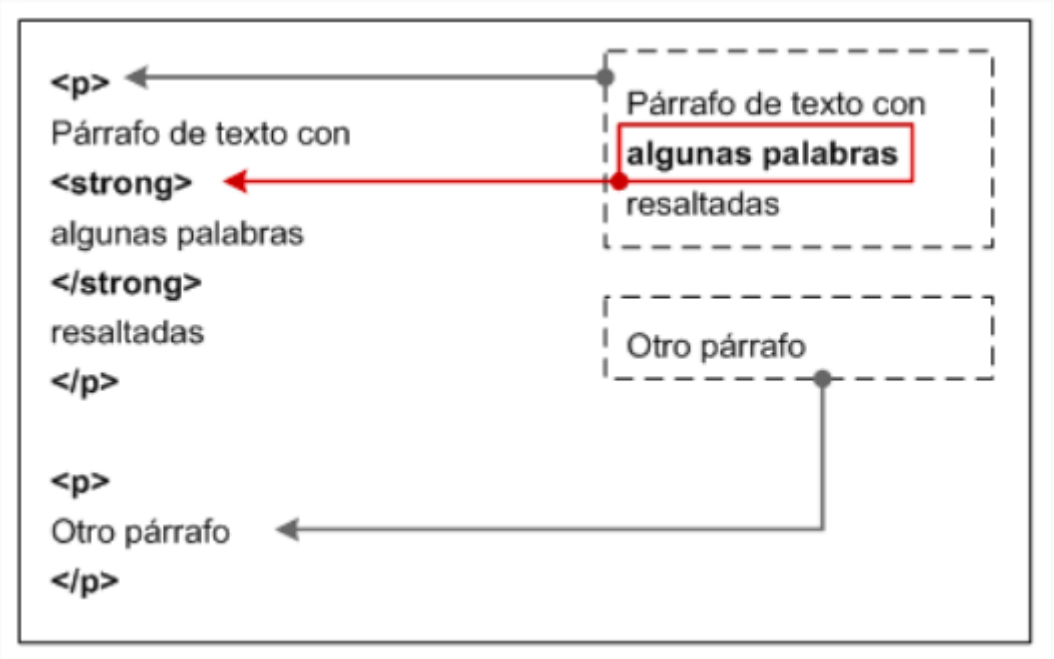
El diseño de cualquier página HTML está compuesto por cajas rectangulares. **CSS permite definir la altura y anchura de cada caja**, el **margen** existente entre cajas y el espacio de **relleno** interior que muestra cada caja. Además, **CSS permite** controlar la forma en la que se visualizan las cajas: se pueden **ocultar**, **desplazar** respecto de su posición original **y fijarlas en una posición específica** dentro del documento.

Como la mayoría de cajas de las páginas web no muestran ningún color de fondo ni ningún borde, no son visibles a simple vista. La siguiente imagen muestra las cajas que forman la página web de https://www.google.es después de forzar a que todas las cajas muestren un borde solido rojo:

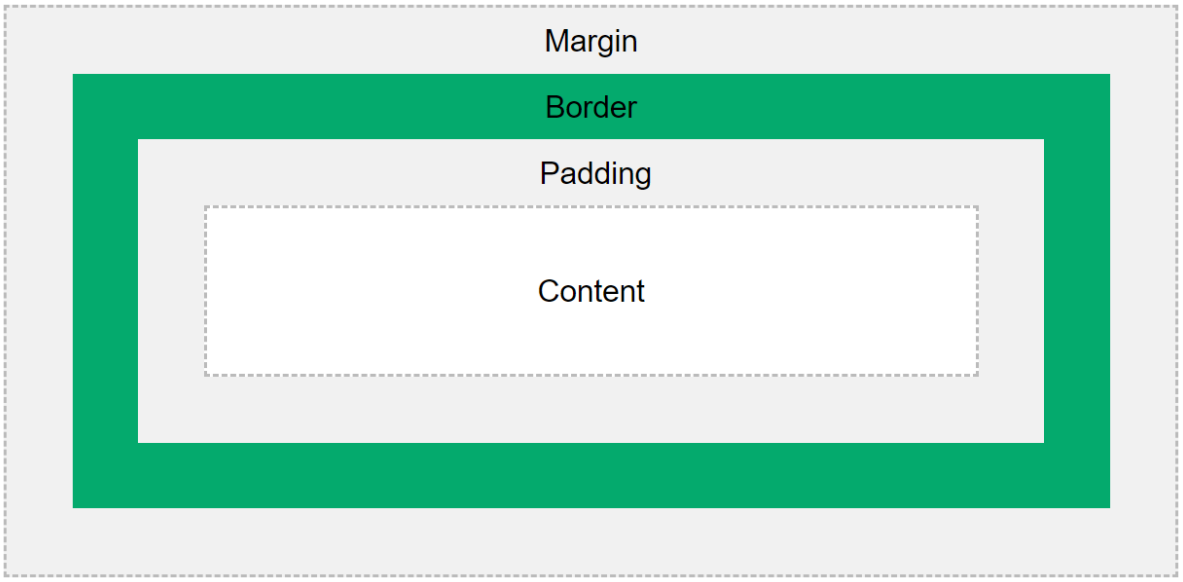
```
* {
    outline: 1px solid red;
}
```



Las cajas de una página se crean automáticamente. Cada vez que se inserta una etiqueta o elemento en la página, se crea una nueva caja rectangular que encierra los contenidos del elemento. El siguiente esquema muestra la creación automática de cajas por parte de HTML para cada elemento definido en el código HTML de la página:



Cada una de las cajas está formada por seis partes, tal y como se muestra en la siguiente imagen:



Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

- **Contenido** (content): se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Relleno** (padding): espacio libre opcional entre el contenido y el borde que lo encierra.
- **Borde** (border): línea que encierra completamente el contenido y su relleno.

- **Imagen de fondo** (background image): imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo** (background color): color que se muestra por detrás del contenido y el espacio de relleno.
- **Margen** (margin): espacio libre entre la caja y las posibles cajas adyacentes.

El relleno y el margen son transparentes, por lo que en el espacio ocupado por el relleno se muestra el color o imagen de fondo (si están definidos) y en el espacio ocupado por el margen se muestra el color o imagen de fondo de su elemento padre (si están definidos). Si ningún elemento padre tiene definido un color o imagen de fondo, se muestra el color o imagen de fondo de la propia página (si están definidos).

Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualiza. No obstante, **si la imagen de fondo no cubre totalmente la caja del elemento o si la imagen tiene zonas transparentes, también se visualiza el color de fondo**. Combinando imágenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.

13.1 Anchura y altura

13.1.1 Anchura - width

La propiedad CSS que controla la anchura de los elementos se denomina **width**.

`width`	Anchura
Valores	`<medida> | <porcentaje> | auto | inherit`
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las filas de tabla y los grupos de filas de tabla
Valor inicial	auto
Descripción	Establece la anchura de un elemento

La propiedad width no admite valores negativos y los valores en porcentaje se calculan a partir de la anchura de su elemento padre.

- El valor inherit indica que la anchura del elemento se hereda de su elemento padre.
- El valor auto, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la anchura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

El siguiente ejemplo establece el valor de la anchura del elemento lateral:

```
#lateral { width: 200px; }
```

```
<div id="lateral">
...
</div>
```

13.1.2 Altura - height

La propiedad CSS que controla la altura de los elementos se denomina **height**.

height	Altura
Valores	<medida> | <porcentaje> | auto | inherit
Se aplica a	Todos los elementos, salvo los elementos en línea que no sean imágenes, las columnas de tabla y los grupos de columnas de tabla
Valor inicial	auto
Descripción	Establece la altura de un elemento

Al igual que sucede con width, la propiedad height no admite valores negativos. Si se indica un porcentaje, se toma como referencia la altura del elemento padre. Si el elemento padre no tiene una altura definida explícitamente, se asigna el valor auto a la altura.

- El valor inherit indica que la altura del elemento se hereda de su elemento padre.
- El valor auto, que es el que se utiliza si no se establece de forma explícita un valor a esta propiedad, indica que el navegador debe calcular automáticamente la altura del elemento, teniendo en cuenta sus contenidos y el sitio disponible en la página.

El siguiente ejemplo establece el valor de la altura del elemento de cabecera:

```
#cabecera { height: 60px; }
```

```
<div id="cabecera">
...
</div>
```

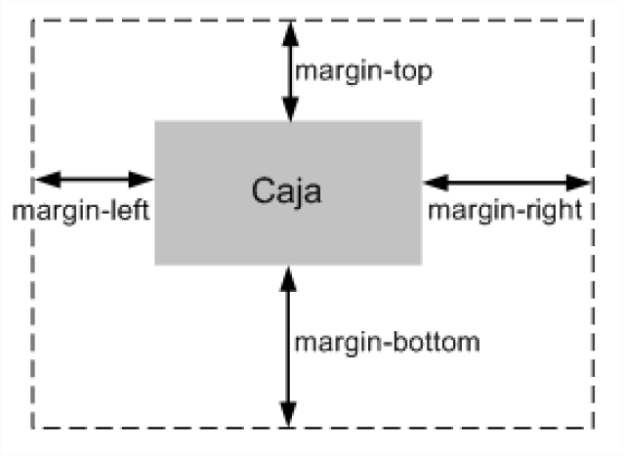
13.2 Margen y relleno

13.2.1 Margen - margin

CSS define cuatro propiedades para controlar cada uno de los márgenes horizontales y verticales de un elemento.

margin-top margin-right margin-bottom margin-left	Margen superior Margen derecho Margen inferior Margen izquierdo
Valores	`<medida> | <porcentaje> | auto | inherit`
Se aplica a	Todos los elementos, salvo margin-top y margin-bottom que sólo se aplican a los elementos de bloque y a las imágenes
Valor inicial	0
Descripción	Establece cada uno de los márgenes horizontales y verticales de un elemento

Cada una de las propiedades establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes:



Las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles (cuando se requiere una precisión total), los em (para hacer diseños que mantengan las proporciones) y los porcentajes (para hacer diseños líquidos o fluidos).

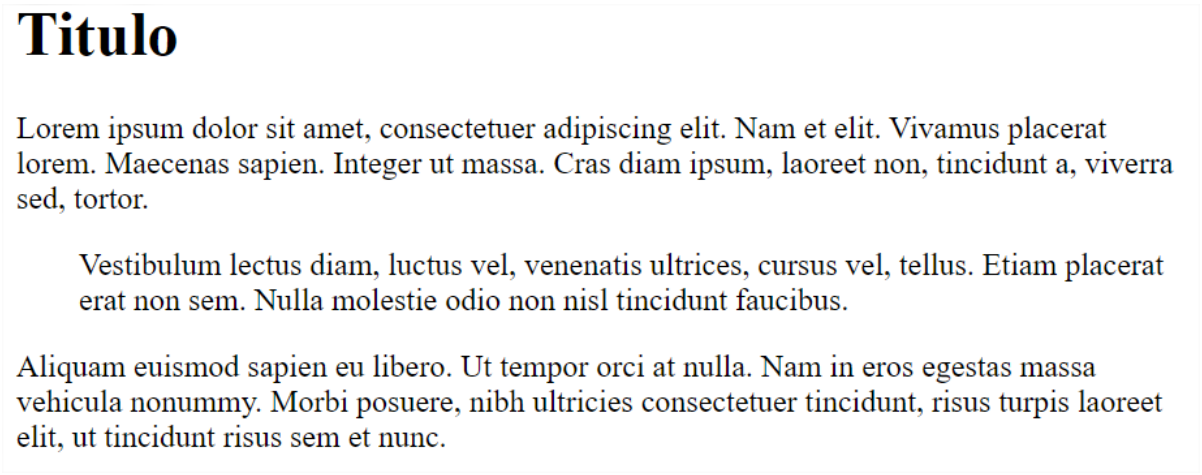
El siguiente ejemplo añade un margen izquierdo al segundo párrafo:

```
.destacado {
    margin-left: 2em;
    }
```

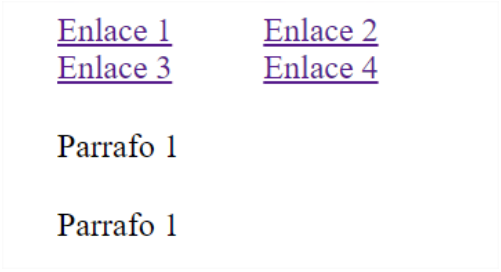
```
    <h1>Titulo</h1>
    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nam et elit.
        Vivamus placerat lorem. Maecenas sapien. Integer ut massa. Cras diam
ipsum,
        laoreet non, tincidunt a, viverra sed, tortor.</p>
    <p class="destacado">Vestibulum lectus diam, luctus vel, venenatis
ultrices,
        cursus vel, tellus. Etiam placerat erat non sem. Nulla molestie odio
non
        nisl tincidunt faucibus.</p>
```

```
    <p>Aliquam euismod sapien eu libero. Ut tempor orci at nulla. Nam in eros
        egestas massa vehicula nonummy. Morbi posuere, nibh ultricies
consectetuer tincidunt,
        risus turpis laoreet elit, ut tincidunt risus sem et nunc.</p>
```

Resultado:



Los márgenes verticales (margin-top y margin-bottom) sólo se pueden aplicar a los elementos de bloque y las imágenes, mientras que los márgenes laterales (margin-left y margin-right) se pueden aplicar a cualquier elemento, tal y como muestra la siguiente imagen:



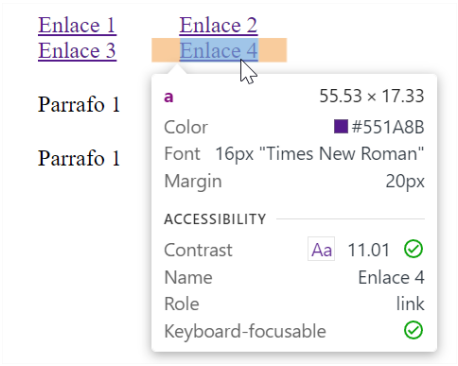

```
a {
    margin: 20px;
    }

p {
    margin: 20px;
}
```

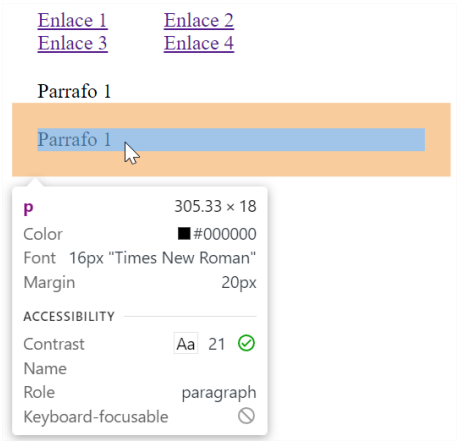
```
    <a href="#">Enlace 1</a>
    <a href="#">Enlace 2</a>
    <br>
    <a href="#">Enlace 3</a>
    <a href="#">Enlace 4</a>

    <p>Parrafo 1</p>
    <p>Parrafo 1</p>
```

La imagen anterior muestra el resultado de aplicar los mismos márgenes a varios enlaces (elementos en línea) y varios párrafos (elementos de bloque). En los elementos en línea los márgenes verticales no tienen ningún efecto, por lo que los enlaces no muestran ninguna separación vertical:

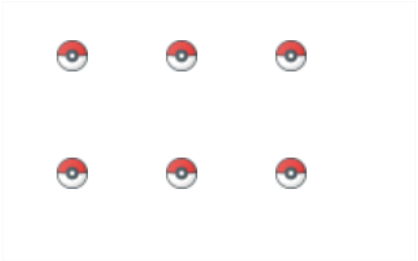


Al contrario de lo que sucede con los párrafos:



Sin embargo, los márgenes laterales funcionan sobre cualquier tipo de elemento, por lo que los enlaces se muestran separados entre sí y los párrafos aumentan su separación con los bordes laterales de su elemento contenedor.

El siguiente ejemplo utiliza el mismo valor en los cuatro márgenes de cada imagen para facilitar su identificación y mejorar el diseño general de la página:



El código CSS del ejemplo anterior se muestra a continuación:

```
div img {
    margin: 20px;
    }
```

La propiedad que permite definir de forma simultánea los cuatro márgenes se

denomina **margin**.

margin	Margen
Valores	(<medida> | <porcentaje> | auto) {1, 4} | inherit
Se aplica a	Todos los elementos salvo algunos casos especiales de elementos mostrados como tablas
Valor inicial	-
Descripción	Establece de forma directa todos los márgenes de un elemento

La notación {1, 4} de la definición anterior significa que la propiedad margin admite entre uno y cuatro valores, con el siguiente significado:

- **Si solo se indica un valor, todos los márgenes tienen ese valor.**
- **Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.**
- **Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.**
- **Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.**

El ejemplo anterior de márgenes se puede reescribir utilizando la propiedad margin:

```
div img {
    margin-top: .5em;
    margin-bottom: .5em;
    margin-left: 1em;
    margin-right: .5em;
}
```

Alternativa directa:

```
div img {
    margin: .5em .5em .5em 1em;
}
```

Otra alternativa:

```
div img {
    margin: .5em;
    margin-left: 1em;
}
```

El comportamiento de los márgenes verticales es más complejo de lo que se puede imaginar. Cuando se juntan dos o más márgenes verticales, se fusionan de forma automática y la altura del nuevo margen será igual a la altura del margen más alto de los que se han fusionado.

Por ejemplo:

```
div {
  margin: 40px 0;
  background: lavender;
}

p {
  margin: 8px 0 25px 0;
  background: yellow;
}
```

```
    <p>El margen inferior de este párrafo está colapsado ....</p>
    <p>... con el margen superior de este párrafo, lo que deja un margen de
<code>25px</code> entre ellos.
    </p>

    <div>
        Este elemento padre contiene dos párrafos!
        <p>Este párrafo tiene un margen de <code>8px</code> entre él y el
texto anterior.</p>
        <p>Mi margen inferior se colapsa con mi padre, produciendo un margen
inferior de <code>40px</code>.</p>
    </div>

    <p>Estoy <code>40px</code> por debajo del elemento de arriba.</p>
```

Resultado:

El margen inferior de este párrafo está colapsado ....

... con el margen superior de este párrafo, lo que deja un margen de `25px` entre ellos.

Este elemento padre contiene dos párrafos!

Este párrafo tiene un margen de `8px` entre él y el texto anterior.

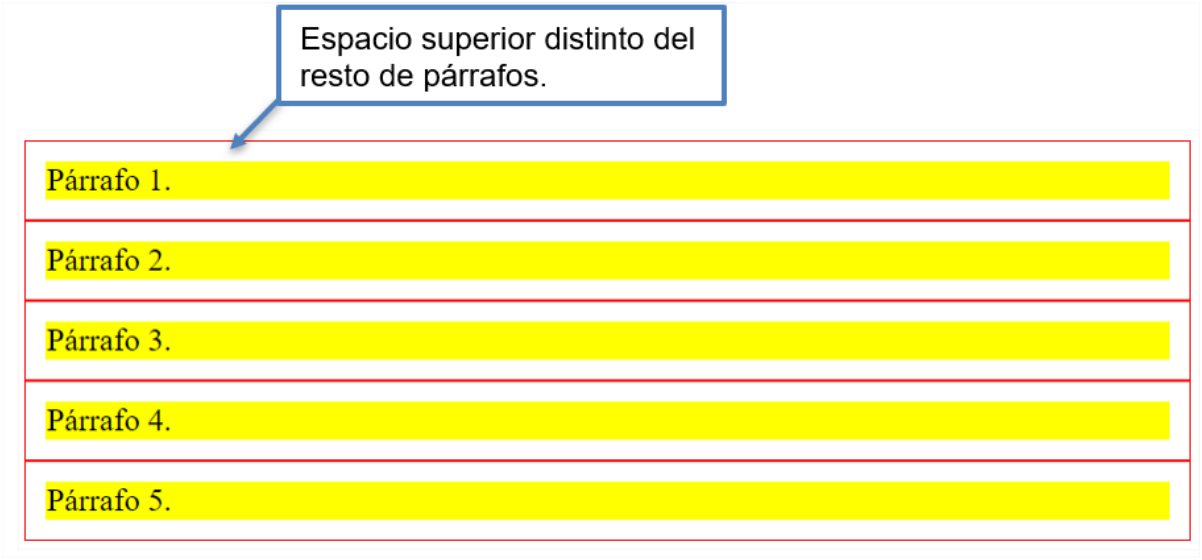
Mi margen inferior se colapsa con mi padre, produciendo un margen inferior de `40px`.

Estoy `40px` por debajo del elemento de arriba.

El espacio entre el primer y segundo párrafo no es 8px + 25px = 33px, sino el valor máximo 25px.

Y entre el último párrafo y el penúltimo no es 25px + 40px =65px, sino el valor máximo 40px.

Aunque en principio puede parecer un comportamiento extraño, la razón por la que se propuso este mecanismo de fusión automática de márgenes verticales es el de dar uniformidad a las páginas web habituales. En una página con varios párrafos, si no se diera este comportamiento y se estableciera un determinado margen a todos los párrafos, el primer párrafo no mostraría un aspecto homogéneo respecto de los demás.



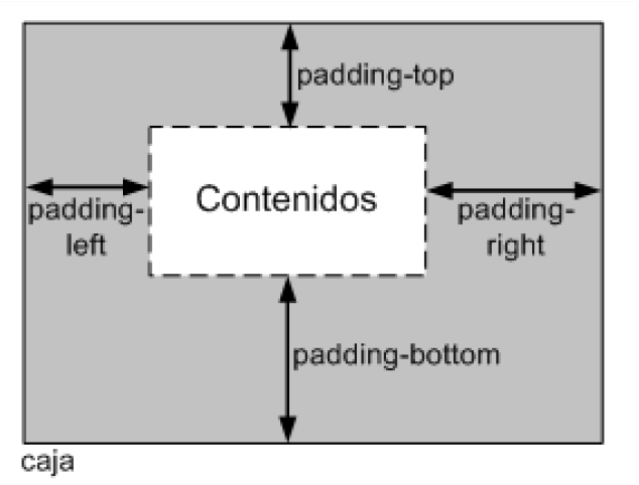
En el caso de un elemento que se encuentra en el interior de otro y sus márgenes se fusionan de forma automática, se puede evitar este comportamiento añadiendo un pequeño relleno (padding: 1px) o un borde (border: 1px solid transparent) al elemento contenedor.

13.2.2 Relleno - padding

CSS define cuatro propiedades para controlar cada uno de los espacios de relleno horizontales y verticales de un elemento.

padding-top **padding-right** **padding-bottom** **padding-left**	Relleno superior Relleno derecho Relleno inferior Relleno izquierdo
Valores	`<medida> | <porcentaje> | inherit`
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	0
Descripción	Establece cada uno de los rellenos horizontales y verticales de un elemento

Cada una de las propiedades **establece la separación entre el lateral de los contenidos y el borde** lateral de la caja:



El siguiente ejemplo muestra la diferencia entre el margen y el relleno de los elementos:

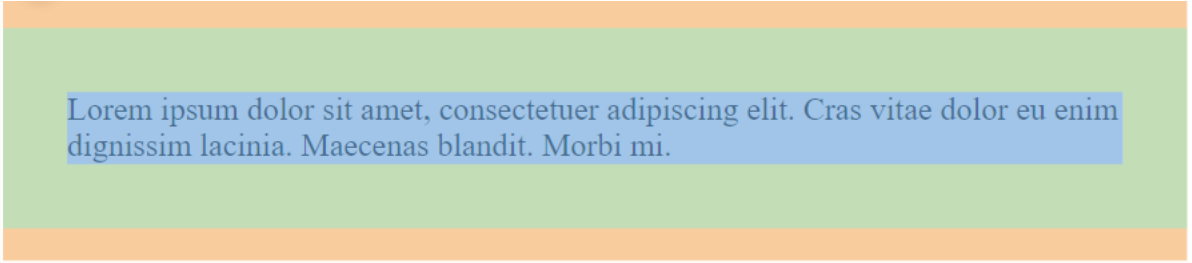
```
.margen {
    margin-top: 2em;
    margin-right: 2em;
    margin-bottom: 2em;
    margin-left: 2em;
}

.relleno {
    padding-top: 2em;
    padding-right: 2em;
    padding-bottom: 2em;
    padding-left: 2em;
}
```

```
<p class="margen">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Cras vitae dolor eu enim dignissim lacinia. Maecenas blandit. Morbi mi.</p>
<p class="relleno">Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Cras vitae dolor eu enim dignissim lacinia. Maecenas blandit. Morbi mi.</p>
```

Resultado:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Cras vitae dolor eu enim dignissim lacinia. Maecenas blandit. Morbi mi.



La propiedad que permite definir de forma simultánea los cuatro márgenes se denomina padding.

padding	Relleno
Valores	(<medida> | <porcentaje>) {1, 4} | inherit
Se aplica a	Todos los elementos excepto algunos elementos de tablas como grupos de cabeceras y grupos de pies de tabla
Valor inicial	-
Descripción	Establece de forma directa todos los rellenos de los elementos

La notación {1, 4} de la definición anterior significa que la propiedad padding admite entre uno y cuatro valores, con el mismo significado que el de la propiedad margin. Ejemplo:

```
body {padding: 2em} /* Todos los rellenos valen 2em */
body {padding: 1em 2em} /* Superior e inferior = 1em, Izquierdo y derecho =
2em */
body {padding: 1em 2em 3em} /* Superior = 1em, derecho = 2em, inferior = 3em,
izquierdo
= 2em */
body {padding: 1em 2em 3em 4em} /* Superior = 1em, derecho = 2em, inferior =
3em,
izquierdo = 4em */
```