

Hojas de estilo en cascada

Sitio: [Aula Virtual IES Aller Ullosa - Informática](http://avn.iesallerullosa.es)
Curso: Deseño de Interfaces Web (2021-2022)
Libro: Hojas de estilo en cascada

Impreso por: Pablo David Rodríguez Jácome
Data: Friday, 5 de November de 2021, 21:35

Táboa de contidos

1. **Introducción a hojas de estilo en cascada.**
2. **Selectores: estilos en línea basados en etiquetas, en clases y en identificadores.**
 - 2.1. Selectores basados en etiquetas
 - 2.2. Selectores basados en clases
 - 2.3. Selectores basados en identificadores
3. **Agrupación de selectores**
4. **Anidamientos de selectores**
5. **Lenguaje de marcas**
6. **Buenas prácticas al escribir CSS**
7. **Atributos. Modelo de cajas**
 - 7.1. Unidades de medida
 - 7.2. Atributos de posición
 - 7.3. Atributos margin
 - 7.4. Atributos padding
 - 7.5. Atributos border
 - 7.6. Atributos del contenido
 - 7.7. Actividad 2.6
8. **Elementos**
 - 8.1. Atributos de fuentes
 - 8.2. Propiedades para textos
 - 8.3. Tablas
 - 8.4. Colores y fondos
 - 8.5. Listas
 - 8.6. Menú vertical con listas
 - 8.7. Menú horizontal con listas
 - 8.8. Pseudo-elementos
 - 8.9. Enlaces
9. **Posicionamiento**
10. **Visualización**
11. **Flexbox**
12. **Box-sizing**
13. **Filtros CSS**
 - 13.1. Modos de fusión
14. **Precedencia de estilos**
15. **Crear y vincular hojas de estilo**
16. **Crear y vincular hojas de estilo en cascada externa**
17. **Diseño web adaptativo**

1. Introducción a hojas de estilo en cascada.

Entre todos los avances que en la última década se han producido en el campo del diseño web, la aparición de CSS (Cascading Style Sheet. Hojas de estilos en cascada) ha sido uno de los más significativos. Su principal ventaja es que permiten separar en el desarrollo de un sitio web lo que es el diseño (apariencia) de lo que son los contenidos (información que se quiere transmitir). Esto tiene como efecto inmediato un desarrollo y mantenimiento más eficiente de sitios web.

Más técnicamente, la filosofía de las CSS se puede resumir así: usar la etiqueta <body> de HTML para definir las estructuras de los contenidos que se muestran en el sitio (encabezados, párrafos, viñetas, etc.) y, luego, en otro archivo o en el <head> del HTML, se define la apariencia de cada página usando el lenguaje de CSS. Esto permite que se puedan cambiar los contenidos que se pongan en el <body> sin afectar a la apariencia definida en el archivo CSS (o en el <head> del HTML), o que se cambie la apariencia en el CSS sin que afecte a nada de lo puesto en el <body> del HTML.

Otra de las grandes ventajas actuales de los CSS es la adaptación de los sitios web a los dispositivos con los que será visualizado. Un sitio web puede ser visto desde un iPad, un móvil con Android, un ordenador personal o cualquiera de los dispositivos disponibles actualmente. La combinación de CSS y HTML permite crear sitios web personalizados para cada dispositivo. Cuando el servidor detecta el dispositivo cliente, éste aplica una hoja de estilos para un mejor visionado. Evidentemente, cada una de las hojas de estilos para cada dispositivo debe haber sido diseñada a propósito por el diseñador, aquí no hay magias.

La organización W3C fue la encargada de estandarizar la versión CSS1. W3C definió CSS como hojas de estilo en cascada ya que es posible que un mismo contenido pueda ser regido por varios archivos CSS que controlen su apariencia. Lo que el estándar CSS1 de W3C pretendía era establecer los criterios por los que se da preferencia a un estilo respecto a otro (por eso lo de cascada). Un mismo elemento, como el encabezamiento <h1> de la página puede estar definido en un archivo CSS externo para aparecer como texto azul; en la propia página, definirlo como texto rojo. En este caso (muy común) el navegador tiene que optar por uno u otro estilo. Por lo tanto, la especificación W3C marca las pautas de preferencia que debe respetar un navegador compatible con CSS.

CSS1 alcanzó el estatus de recomendación por la W3C en el año 1996. Las reglas CSS1 tienen un soporte adecuado en prácticamente todos los navegadores modernos más conocidos. Las reglas CSS 2 alcanzaron el estatus de recomendación en el año 1998. En junio de 2011, el módulo de colores de CSS 3 Color ha sido publicado.

El objetivo de que W3C estandarice las CSS, HTML o cualquier otro lenguaje de Internet, es garantizar que el creador de un sitio web no tiene que hacer uno a propósito (ad hoc) para cada uno de los navegadores web existentes (Explorer, Firefox, Chrome, Opera, etc.) y, además, que los usuarios no vean un sitio web con apariencia diferente dependiendo del navegador que empleen.

2. Selectores: estilos en línea basados en etiquetas, en clases y en identificadores.

Una hoja de estilo CSS está formada por reglas que indican la manera en la que se visualizará la página (reglas de estilo). Cada regla está compuesta de selectores lo cuales indican a qué elemento o parte de una página se aplica un determinado estilo.

2.1. Selectores basados en etiquetas

Para dar los primeros pasos en la definición de reglas de estilo, la manera más sencilla es usar las propias etiquetas HTML como selectores. Esto consiste en asociar a cada etiqueta una declaración del estilo que se le aplica al selector (etiqueta HTML). Las declaraciones tienen esta forma:

```
selector { atributo:valor }
```

El atributo hace referencia a la característica que se quiere modificar de la etiqueta, por ejemplo color. El valor hace referencia a la instancia del atributo, por ejemplo, blue.. Así, por ejemplo, h1 podría ser un selector para aplicar un estilo a la etiqueta <h1>. Para indicar entonces el estilo de <h1> se pondría:

```
h1 {color:blue}
```

Los selectores se escriben omitiendo las llaves < >, es decir, simplemente h1, h2, etc. La declaración {atributo:valor} ha de ir encerrada en llaves { }.

A cualquier etiqueta HTML se le puede asignar un estilo, pero la descripción de las reglas debe ajustarse a la sintaxis definida anteriormente (según la especificación CSS). Si un navegador encuentra un selector cuya sintaxis no comprende, éste ignorará la declaración entera y continúa con la siguiente, con independencia de si el error afecta a la propiedad, al valor o a toda la descripción.

CSS contempla sintaxis para definir atributos para varios selectores y selectores con varios atributos.

```
Selector 1, Selector2, {atributo1:valor 1; atributo2:valor2}
```

Así, por ejemplo, si el mismo estilo se le quiere aplicar a dos selectores distintos, la sintaxis sería:

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

y si al mismo selector se le quiere aplicar atributos diferentes:

```
h1 {color: blue; background-color:red}
```

La propiedad background-color hace referencia al color de fondo del texto que por defecto es como el de la página.

Una vez conocida la sintaxis de las reglas, la siguiente pregunta sería ¿dónde se debe poner esas declaraciones para que el navegador las lea y las interprete? Una respuesta válida es en la cabecera <head> </head>. Todas las declaraciones basadas en etiquetas se incluyen en el <head> del fichero HTML entre etiquetas <style></style>. Así, por ejemplo, las declaraciones del ejemplo anterior quedarían así:

```
<head>
<title></title>
<style>
h1 {color: blue; background-color:red }
</style>
</head>
```

Cuando en navegador lee la cabecera del HTML interpreta que todas las etiquetas <h1> incluidas en el <body> tendrán color azul y color de fondo rojo.

Actividad:

Cree un fichero HTML con dos textos, uno entre etiquetas <h1 ></h1> y otro entre <h2></h2>. Luego incluya en el <head> un estilo que afecte al color de las etiquetas h1 (color) y al color de fondo (background-color) de los h2

2.2. Selectores basados en clases

Los selectores basados en etiquetas tienen un uso muy claro, lo que las hace fáciles de entender. Sin embargo, desde el punto de vista de la flexibilidad y la reutilización, esta alternativa no es muy ventajosa.

Un ejemplo que muestra esa falta de flexibilidad es si se desea establecer diferentes estilos según el tipo de párrafo que se ponga, de manera que se distinga el encabezado de la página del resto del texto. Con lo visto en la sección anterior, si se hace una declaración del tipo `(color:blue)` siempre que aparezca un texto entre etiquetas `<p></p>` éste se mostrará en color azul. Pero, si lo que se desea es que unos párrafos respondan a una regla de estilo y otros a otra esta alternativa no es válida. Es necesario usar selectores basados en clases.

Mediante las clases se pueden definir estilos abstractos, es decir, que no estén asociados directamente a una etiqueta HTML. Las clases permiten aplicar estilos a etiquetas HTML, con el mismo efecto que usando selectores de etiquetas, pero también a cualquier otro elemento de la página.

Hay diferentes tipos de clases: unas asociadas directamente en una etiqueta HTML (por ejemplo `h1.verde (color:green)`) y otras más genéricas que se pueden aplicar a cualquier etiqueta (por ejemplo, `.citas (color:grey)`).

Las clases asociadas a etiquetas HTML se definen con el

Nombreetiqueta.nombreclase {atributo:valor; atributo:valor; ... }

Esta alternativa es más potente que la vista con selectores basados en etiquetas, ya que permite aplicar a las mismas etiquetas diferente estilo.

```
<head>
<style>
h1.roja {color: red}
h1.verde {color: green}
h1.azul {color: blue}
</style>
</head>
```

Para indicar en cada etiqueta `<h1>` qué estilo se le quiere aplicar se usa el atributo `class` de la siguiente manera:

```
<body>
<h1 class="roja"> Un encabezamiento rojo </h1>
<h1 class="azul"> ahora azul </h1>
<h1 class="verde"> Y ahora verde </h1>
</body>
```

El resultado quedaría: textos de tamaño grande pero de diferente color (el cambio de color se apreciará al ejecutar el código en el navegador).

Un encabezamiento rojo

ahora azul

Y ahora verde

Las clases más genéricas no se aplican a ninguna etiqueta HTML, por lo que en su descripción se emite en el selector el nombre de ninguna etiqueta.

. nombreclase {atributo:valor; atributo:valor; ... }

Por ejemplo:

`.verde {color:green; }`

Una clase así definida puede aplicarse a cualquier elemento de la página. Del ejemplo siguiente, la primera declaración se aplica a todo el párrafo, la segunda a todo el encabezado `<h1>` y la tercera a todo el bloque `<div>`:

```
<p class="verde"> ...
<h1 class="verde"> ...
<div class="verde"> ...
```

Con la siguiente declaración:

```
<head>
<style>
.roja {color: red}
.verde {color: green}
.azul {color: blue}
</style>
</head>
```

Y aplicando las clases a las diferentes etiquetas `<h1>`:

```
<body>
<h1 class="roja">Un encabezamiento rojo</h1>
<h1 class ="azul ">ahora azul </h1>
<h1 class = "verde">Y ahora verde</h1>
</body>
```

Se obtienen un resultado idéntico al anterior. Sin embargo, con esta solución, no solo se limita aplicar estas clases a las etiquetas `<h1>` sino que se pueden aplicar a cualquier otra, por ejemplo a una etiqueta `<p>`.

```
<p class="roja">Aquí está el párrafo en rojo </p>
```

Si esa línea HTML se incluye en el `<body>` del ejemplo anterior quedaría:

Un encabezamiento rojo

ahora azul

Y ahora verde

Aquí está el párrafo en rojo

Una característica muy interesante del uso de clases es que se puede asignar más de una clase o una misma etiqueta, siempre y cuando no hay conflicto entre ellas. Por ejemplo, con la siguiente definición:

```
<head>
<style>
.textorojo {color: red}
.fondoazul {background-color: blue }
</style>
</head>
```

Se podría hacer el siguiente contenido en el `<body>`:

```
<h3 class="textorojo fondoazul">título en rojo, fondo azul</h3>
<p class="textorojo">color en rojo; fondo: el que herede de la página</p>
```

Actividad:

Cree un fichero HTML con dos textos, uno entre etiquetas `<h1></h1>` y otro entre `<p></p>`. Luego incluya en el `<head>` estilos con clases abstractas que muestren: el texto en `<h1 >` en verde con fondo azul y el texto de `<p>` en azul con fondo verde. Resuelva la actividad de dos maneras:

a. La primera, usando una declaración de estilos con 4 etiquetas (2 para los colores y dos para los fondos).

b. La segunda usando dos etiquetas: una que se puede llamar `.textoverdefondoazul` y otra que se puede llamar `.textoazulfondoverde`

2.3. Selectores basados en identificadores

A modo de resumen de esta sección anterior, las clases definen propiedades que pueden ser compartidas por uno o varios elementos. Esto hace que una clase, por ejemplo .roja vista anteriormente, puede ser usada en un elemento `<h1 class="roja"></h1>` y en un `<p class="roja"></p>`. Es decir, las clases se pueden utilizar en varios elementos:

Concluido esto, en esta sección se explicarán los selectores basados en identificadores, los cuales tienen una función y sintaxis muy parecida, pero que se diferencian en algo muy sutil: los identificadores solo se pueden usar en un único elemento.

Un selector basado en identificador se define dentro de las etiquetas `<style></style>` con la siguiente sintaxis:

Nombreetiqueta#nombreclase {atributo:valor; atributo:valor; ... }

#nombreclase {atributo:valor; atributo:valor; ... }

Como puede apreciarse, esta declaración es idéntica a la usada para definir selectores de clase. La única diferencia es que en vez de usar un punto "." se usa una almohadilla "#". Sin embargo, el significado, la semántica de ambas expresiones es muy parecida.

Luego, para indicar en cada etiqueta `<h1>` qué estilo se le quiere aplicar se usa el atributo `id` en la etiqueta (de la misma manera que se usa `class` para las clases). Así, el siguiente ejemplo muestra una definición de estilos usando selectores de identificador.

```
<head>
<style>
#rojo { color:red; }
</style>
</head>
<body>
<p id="rojo" > el párrafo va en rojo </p>
</body>
```

Este código el navegador lo interpreta mostrando en color rojo "el párrafo va en rojo".

En el ejemplo anterior, si cambiamos `id` por `class` y "#" por "." en vez de aplicar selectores de identificador aplicamos selectores de clase. Aparentemente no hay diferencia entre identificadores y clases. Sin embargo, si que las hay, muy sutiles, pero importantes.

La diferencia entre identificadores y clases es la misma que entre clases y objetos en un modelo orientado a objetos. A grandes rasgos **las clases definen un patrón que deben cumplir todos los objetos de la clase. Cada objeto se identifica con un identificador único que lo diferencia de los demás objetos de esa clase.**

En CSS las clases se pueden usar en uno o varios elementos. Por ejemplo, en una etiqueta `<h1>` y en una `<h2>` y en otra `<h1>` diferente y en una `<p>`. Sin embargo, y esta es la diferencia, los identificadores solo se deben usar en un único elemento. Esto es porque un identificador es como si hiciese referencia a un objeto de estilo y los objetos son únicos, por tanto solo un elemento puede "tener" ese objeto de estilo.

Un ejemplo que muestra la diferencia de uso es el siguiente: La clases son habituales usarles de esta manera, que muestra cómo la clase `textorojo` se usa en dos elementos `<div>`:

```
<div class="textorojo"></div>
```

El texto hereda las propiedades de una clase `textorojo`

```
<div class="textorojo"></div>
```

Se vuelve a heredar las propiedades de la misma clase `textorojo`

```
<div class="textonegrita" ></div>
```

El texto hereda las propiedades de las clases `textonegrita`.

Sin embargo, los identificadores se suelen usar en casos como el siguiente en donde los identificadores `cabecera`, `contenidos` y `piepagina` definen el estilo de esas tres partes de una página web.

Los tres objetos `<div>` son asociados a tres identificadores diferentes ya que no tiene sentido que esos identificadores se repitan en varios elementos (no tiene sentido que una página tenga por ejemplo dos o más elementos `<div>` con un estilo tipo `cabecera` en una misma página web).

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

```
<div id="contenido"></div>
```

```
<div id="piepagina"></div>
```

En definitiva, las clases se usan cuando el estilo se quiere aplicar a más de un elemento, mientras que los identificadores se definen cuando lo que se busca es "exclusividad" ya que solo será aplicada a un elemento. Una práctica habitual de los identificadores es usarlos para nombrar los bloques o secciones principales de un sitio web. El resto de estilos se hacen con clases.

Realmente, en muchos de los navegadores actuales, si se usa un identificador en varios elementos estos se interpretan y devuelven un resultado idéntico al que se devuelve si hiciese con clases. Sin embargo, son "buenas prácticas" de diseñador usar los identificadores y las clases es su momento, con vistas a seguir un estándar de uso y se también para que el diseñador se pueda beneficiar de las ventajas de los identificadores. Por ejemplo, CSS permite dar como valor un identificador al atributo `href` de la etiqueta `<a>` de la siguiente manera:

```
<head>
<style>
#cabecera{
background: #CCC;
border: 1px solid #093;
margin:10px 12px 20px 15px; }
</style>
</head>
<body>
<div id="cabecera"> Aquí está la cabecera </div>
<a href="#cabecera"> Ir a la cabecera </a>
</body>
```

En este otro ejemplo la etiqueta `<a>` define un enlace al elemento que usa el identificador "cabecera" que es un `<div>`. Si se usan clases o se usa un identificador en varios elementos (no recomendable) el navegador no sabría a qué elemento definido con estilo "cabecera" saltar. Esa es solo una de las muchas ventajas que tiene usar identificadores según se recomienda en la especificación CSS.

Actividad:

Cree un fichero HTML que defina un selector de clase y un selector de identificador. Pruebe a usar `ID` y `CLASS` en uno y varios elementos HTML y compruebe su efecto. ¿Qué pasa si no se cumple la especificación y se repite un identificador en varios elementos? ¿cómo responde el navegador? ¿Y en otro navegador, responde igual?

3. Agrupación de selectores

El término agrupamiento hace referencia a la manera en la que se pueden escribir las reglas de estilo (selectores) para conseguir un CSS más claro y fácil de entender. De esta manera, los errores son más fáciles de detectar y corregir. El uso de agrupamientos, como ocurría con el uso de los id y class en la sección anterior, responden a buenas prácticas en la especificación de CSS. Como se verá, algunas de las reglas de agrupamiento han sido ya comentadas anteriormente.

Cualquier selector se puede agrupar siguiendo esta sintaxis:

Selector1, Selector2, {atributo1: valor1; atributo2: valor2; ... }

De esta manera se puede aplicar el mismo estilo a un conjunto de selectores al mismo tiempo. Por ejemplo, si se quiere aplicar un tipo de fuente Arial u etiquetas <h1>, <p> y <h3>, la regla sería:

```
h1,p,h3 {font-family:arial}
```

La versión menos optimizada de esa misma regla sería:

```
h1 { font-family: arial; }
```

```
p1 {font-family: arial }
```

```
h3 {Font-family : arial }
```

De la misma manera se podría hacer para selectores de clase, pero teniendo en cuenta que si son abstractos es necesario primero poner un ".". El siguiente ejemplo le asigna un color en hexadecimal (#0000FF) a tres clases: mensaje, énfasis y subtítulo.

```
.mensaje, .subtitulo, .enfasis {color:#0000FF}
```

Lo mismo se puede hacer para selectores de identificador, pero recordando que se usa la almohadilla. El siguiente ejemplo asigna a los identificadores un color verde (#00FF00):

```
#cabecera, #tpiepagina {color:#00FF00}
```

Como se ha comentado antes, las agrupaciones tienen como objetivo simplificar y dejar más claro un CSS. Por lo tanto, también se pueden combinar diferentes tipos de selectores para agrupar según estilos. Así, por ejemplo, la clase .cabecera, la etiqueta h2 y el identificador #micolor comparten el mismo color en hexadecimal (#FF0000):

```
.cabecera, h2, #micolor {color:#FF0000}
```


4. Anidamientos de selectores

Los selectores se pueden anidar con el fin de conseguir estilos más concretos y definidos. Este anidamiento es lo que se llama en CSS selectores contextuales que permiten aplicar un estilo a un elemento dependiendo de los elementos que tenga alrededor.

Selector anidado común

Se usa para crear reglas sobre elementos que están rodeados de otros elementos. La sintaxis general de este tipo de anidamiento para dos selectores es la siguiente:

```
SelectorX SelectorY { atributo1:valor1; atributo2:valor2; ... }
```

Observe que entre ambos selectores hay un espacio en blanco.

Este tipo de anidamiento es útil cuando, por ejemplo, se desea ver en rojo un texto en negrita siempre y cuando esté incluido dentro de una etiqueta <h1> y en cursiva <i> (aunque entre medias haya otras etiquetas).

Si se definen los selectores de esta manera:

```
b {color:red}
```

el siguiente código HTML visualizará ambos textos en rojo, con independencia de la etiqueta <h1> y <i>:

```
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y rojo </b></i> </h1>
<b> Un texto en negrita y rojo </b>
</body>
```

Sin embargo, usando un anidamiento en la definición de la regla se consigue el efecto deseado:

```
<head>
<style>
h1 i b {color:red}
</style>
</head>
```

En principio no hay límite en el número de anidamiento que se pueden hacer, sin embargo, no es recomendable un anidamiento de más de 4 o 5 por motivos de complejidad a la hora de interpretar el CSS, tanto por el navegador como por los diseñadores. Por otro lado, el anidamiento en los ejemplos de arriba se ha hecho con selectores basados en etiqueta, pero también pueden usarse con class, id, o combinaciones.

Enlazando con los agrupamientos vistos anteriormente, este tipo de selectores se pueden agrupar. Por ejemplo, si suponemos que hemos definidos varias reglas con anidamiento h1 i b y h1 b, el agrupamiento sería así:

```
h1 i b, h1 b { color: red; }
```

que equivaldría a esto:

```
h1 i b {color:red}
```

```
h1 b {color:red}
```

Anidamiento de selectores hijos

El anidamiento común explicado anteriormente se comporta igual si las etiquetas están consecutivas o si hay etiquetas intermedias. Por ejemplo, el siguiente anidamiento mostraría los dos textos en verde.

```
<head>
<style>
h1 b {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y rojo </b></i> </h1>
<h1><b> Un texto en negrita y rojo </b></h1>
</body>
```

El motivo es que, para el navegador, tanto <h1><i><h> como <h1> cumplen el anidamiento definido h1 b {color:red}.

Si lo que se desea es restringir que las etiquetas, además de estar en el mismo contexto, estén seguidas unas de otras, entonces la sintaxis que se tiene que usar en la definición es la siguiente (para anidamiento de dos selectores):

```
selectorX > selector Y {atributo1:valor1; atributo2:valor2; ... }
```

De esta manera, el siguiente código mostrará en rojo solo el texto que tiene una etiqueta dentro de <h1> sin ninguna entre medias.

```
<head>
<style>
h1>b {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y negro</b></i> </h1>
<h1><b> Un texto en h1, negrita y rojo </b></h1>
</body>
```

El siguiente código muestra un ejemplo que incluye a tres etiquetas:

```
<head>
<style>
h1>b i {color:red}
</style>
</head>
<body>
<h1><i><b> Un texto h1 en negrita, cursiva y negro</b></i> </h1>
<h1><b><i><u> Texto en h1, negrita, cursiva, rojo y subrayado </u></i></b></h1>
</body>
```

Al igual que el resto de anidamientos, aunque los ejemplo se ha hecho con selectores basados en etiqueta también pueden usarse con class, id, o combinaciones.

Anidamiento de selectores adyacentes

Este tipo de anidamiento se usa cuando se quiere aplicar un estilo a un elemento que tiene adyacente (al lado) a otro elemento en el mismo nivel de anidamiento en HTML. La sintaxis es la siguiente (para dos selectores):

```
selectorX + SelectorY {propiedad1:valor1; propiedad2:valor2; ... }
```

El siguiente ejemplo muestra este estilo de anidamiento sobre dos etiquetas <i> y <h>. Solo se aplicará el estilo sobre si hay una etiqueta adyacente <i>.

En este ejemplo en concreto, la palabra advertencia será la única que aparezca en rojo.

```
<head>
<style>
i+b {color:red}
</style>
</head>
<body>
<p> <i>Nota</i>, esto es una <b>advertencia</b> </p>
<b> Leer detenidamente </b>
</body>
```

Al igual que el resto de anidamientos, aunque los ejemplo se ha hecho con selectores basados en etiqueta también pueden usarse con class, id, o combinaciones.

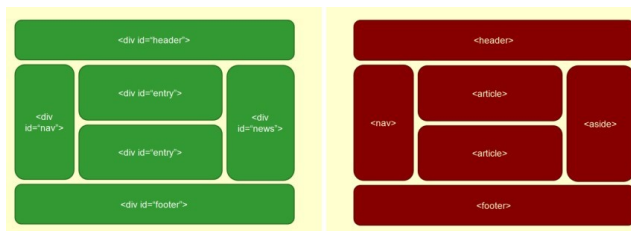
Actividad:








Interprete qué hace el siguiente código. ¿De qué color parecerán los elementos de la lista? ¿De qué tamaño? Una vez interpretado compruebe el resultado en el navegador.

```
<head>
<style>
i+b {color:red}
.nivel1 {color:green}
.nivel2 {color:blue}
ul ul .nivel2 { font-size: x-small }
ul ul li {font-size: x-small; color:red}
</style>
</head>
<body>
<h1><p> <i>Título</i>: <b> Anidamiento y agrupamiento </b> de selectores </p></h1>
<ul>
<li class="nivel1"> Agrupamiento</li>
<li class="nivel1"> Anidamiento </li>
<ul>
<li class="nivel2"> Anidamiento de hijos</li>
<li class="nivel2"> Anidamiento de adyacentes </li>
<li> Anidamiento común</li>
</ul>
</ul>
</body>
```

5. Lenguaje de marcas

Comparación HTML con HTML5



Elemento	Descripción
<body>	Representa el contenido principal de un documento HTML. Solo hay un elemento <body> en un documento.
<section> 	Define una sección en un documento.
<nav> 	Define una sección que solamente contiene enlaces de navegación
<article> 	Define contenido autónomo que podría existir independientemente del resto del contenido.
<aside> 	Define algunos contenidos vagamente relacionados con el resto del contenido de la página. Si es removido, el contenido restante seguirá teniendo sentido
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	Los elemento de cabecera implementan seis niveles de cabeceras de documentos; <h1> es la de mayor y <h6> es la de menor importancia. Un elemento de cabecera describe brevemente el tema de la sección que introduce.
<header> 	Define la cabecera de una página o sección. Usualmente contiene un logotipo, el título del sitio Web y una tabla de navegación de contenidos.
<footer> 	Define el pie de una página o sección. Usualmente contiene un mensaje de derechos de autoría, algunos enlaces a información legal o direcciones para dar información de retroalimentación.
<address>	Define una sección que contiene información de contacto.
<main> 	Define el contenido principal o importante en el documento. Solamente existe un elemento <main> en el documento.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <header style="height: 100px; width: 80%; position: relative; margin:0px auto; /*centra el div en la página */ background-color: cadetblue;" >
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Consectetur perspicatis temporibus blanditiis exercitationem architecto illo molestiae doloreque dolores ullam ab ut vero, sed molestias assumenda repellat iusto accusantium nisi placeat.
  </header>
  <nav style="height: 400px; width: 10%; position:relative; left: 10%; background-color: coral; float: left;">
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Alias nesciunt suscipit at nobis laboriosam tempore beatae deserunt quo enim eius. Quod eius dicta adipisci magni quaerat commodi, at minus impedit.
  </nav>
  <aside style="height: 400px; width: 10%; position:relative; background-color:darkslategray; float: right; right: 10%;">
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Illo inventore nisi dicta laudantium eos, error id facilis deserunt eligendi ex, incidunt aliquam architecto! Consequatur fuga cumque vitae qui voluptas voluptatem.
  </aside>
  <article style="height: 200px; width: 60%; position: relative; left: 10%; background-color: darksalmon; display: inline-block;">
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Nobis, ex hic temporibus nam earum porro. Mollitia cum dicta dolore. Dignissimos officia voluptatum ullam alias officiis ipsam ipsum explicabo dolores! Nobis!</article>
  <article style="height: 200px; width: 60%; position: relative; left: 10%; background-color: darksalmon; display: inline-block;">
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Aliquid pariatur ea esse inventore ut blanditiis veritatis, minus corporis. Debitis, adipisci autem porro dolorem pariatur sit! Aspernatur inventore eaque repellat deserunt.</article>
  <footer style="height: 100px; width: 80%; position: relative; background-color: cadetblue; margin:0px auto;">Lorem ipsum dolor sit amet consectetur adipisicing elit. Laborum dolore ducimus aliquid ullam iusto nostrum, ex, reprehenderit nam atque quaerat cum fugiat provident. Harum corrupti nobis, voluptatum voluptate dolorem min
  </footer>
</body>
</html>
```

6. Buenas prácticas al escribir CSS

Hasta el momento los ejemplos usados en este capítulo eran muy básicos, por tanto, no necesitaban de un número elevado de reglas. Sin embargo, en un trabajo real, las reglas CSS se multiplican para cualquier cosa que se quiera hacer de manera profesional en una web. Si a eso se le suma que cada selector puede tener del orden de 5 o más atributos, el CSS resultante puede ser ilegible si no se estructura con cuidado.

Como se comprobará en ejemplos posteriores, el gran número de reglas que puede tener un CSS necesita que el desarrollador haga un esfuerzo a la hora de escribir los selectores con el fin de que sea más fácil hacer modificaciones posteriores. Las siguientes recomendaciones no están definidas en el estándar W3C, son solo buenas prácticas que la mayoría de los desarrolladores de CSS suelen tener en cuenta. Seguir estas prácticas no solo ayuda a un desarrollo propio más claro, sino que también ayudan a entender mejor desarrollo de otros autores.

Los selectores se nombran en minúsculas, nunca empezando por caracteres especiales o numéricos: como en cualquier otro lenguaje de etiquetas, hacerlo así da más claridad al selector (además de que la especificación CSS no lo permite para nombre de selectores de clase e identificadores). CSS en principio no distingue entre mayúsculas y minúsculas, salvo en los nombres de selectores de clase e identificadores, pero se eliminan errores si todo se pone en minúsculas. Por último, aunque en las últimas versiones se permite usar “_” en los nombres, es su lugar (como luego veremos) es preferible usar “-”, ya que no todos los navegadores soportan “_”.

El nombre de los selectores debe ser específico y claro, para que tenga una mayor capacidad expresiva: los ejemplos anteriores se podrían sustituir por lista-nivel1 {color:green}, de esa manera se entiende mejor que lo que se pretende es dar un estilo para un elemento de una lista que está a nivel1.

El nombre de las clases e identificadores no debe describir una característica visual, como color, tamaño o posición: es mejor no usar nombre asociado a color, tamaño o posición porque hace que el selector soporte peor los cambios. Si a una regla se le llama .rojo{color:rojo}, entonces si por cualquier motivo (que los hay) se cambia el color de la clase, también se debería cambiar el del selector, con los problemas que eso lleva al tener que actualizar todas las referencias a esa clase en el HTML.

Los nombres deben seguir más una visión semántica que estructural: por el mismo criterio de facilitar los cambios, no se deben usar nombre de selectores según la localización si no se dan otras alternativas en el mismo CSS. Por ejemplo, si se usa un selector de clase menú-izquierda(...) para referirse a un menú situado a la izquierda. Si por cualquier motivo se quiere cambiar a la derecha, entonces hay que cambiar también el nombre del selector y de las referencias HTML. Es mejor usar menu-navegacion {...} para definir este selector y no asociarlo a la izquierda. Otra cosa, sería si se desea hacer un menú a la izquierda y otro a la derecha, y que sea el usuario el que seleccione el que más le gusta (como ocurre en CMS como Joomla).

Una alternativa semántica define los selectores según su función y no la estructura donde estará alojada. Por ejemplo, a un <div> es preferible asociarle una clase llamada .main que una llamada .left-content (contenidos de la izquierda) ya que si por cualquier motivo se cambia de lugar se tienen que cambiar sus propiedades y las referencias en el HTML.

Si en cualquier caso el desarrollador piensa que es necesario definir una clase única y exclusivamente para alinear una imagen o un párrafo y llamarla con esa acción, lo importante es documentarlo apropiadamente, para que él u otros desarrolladores no tengan problemas en el futuro. En CSS los comentarios se ponen como en Lenguaje C:

```
/* texto del comentario */
```

Separa las palabras mediante guiones o mayúsculas: así se le da más claridad a los nombres de los selectores. Por ejemplo, menu-superior en vez de menu-navegacion lo hace más legible.

No hacer uso excesivo de clases: esto es útil ya que, a menudo, es más sencillo utilizar selectores contextuales o anidar selectores que no alteren el HTML, o incluso usar selectores de etiquetas para conseguir lo mismo. Los selectores de clase se suelen dejar para las partes más relevantes de la estructura.

Por ejemplo, en vez de usar:

```
<div class="main">
  <div class="main-title"> ... </div>
  <div class="main-paragraph"> ... </div>
</div>
```

Usar esta estructura ya que lo que se busca es el mismo efecto:

```
<div class="main">
  <h1>... </h1>
  <p> .... </p>
</div>
```

Agrupar las reglas según su selector siempre que sea posible: cuando hay varias reglas con el mismo selector (sea del tipo que sea) es interesante agruparlos unas debajo de otras. Por ejemplo:

```
table {border:double}
table.miembros {border:solid}
table.empleados {border:grove}
```

Al principio de un CSS es aconsejable definir los selectores de etiquetas: además se usan comentarios para dejar claro cuál es la parte que define los selectores de etiquetas y cuál es la parte que definen las clases y otros elementos. Por ejemplo:

```
/* Etiquetas HTML */

html {font-family:arial, verdana, sans serif; font-size:13px;}
h1, h2, h3, h4, h5, h6, form, input, text-area{
  border:0; padding:0; margin:0;
  font-family:arial;}
h1 { font-size: 24px; color: 1000000; }
h2 { font-size: 18px; color: 1666666; }

/* FIN Etiquetas HTML */
```

Estructurar visualmente los atributos: si un elemento solo tiene tres atributos se pueden poner en la misma línea. Pero si hay más, se ponen en líneas diferentes sangrados con tabuladores. En el ejemplo anterior se puede ver este uso.

Estas son solo algunas propuestas de buenas prácticas. Sin embargo, conforme el diseñador profundiza en el desarrollo de CSS adquirirá muchas otras que darán más legibilidad a sus CSS.

7. Atributos. Modelo de cajas

Hasta el momento, todos los ejemplos anteriores estaban relacionados con la sintaxis de CSS. Por ello, en los ejemplos siempre se ha jugado con propiedades muy sencillas y fáciles de entender, como el color, la fuente (color), (font-family) o el tamaño de la fuente (font-size). Estas tres propiedades (color, font-family y font-size) se llaman a tributos.

La potencia de los CSS radica principalmente en la gran cantidad de atributos que se pueden usar para dar estilo a las páginas web. Estos van desde la posición que puede ocupar un determinado elemento hasta la colocación de una imagen de fondo.

En esta sección se explicarán aspectos básicos de localización de elementos en las páginas y el significado de los atributos que definen esas distancias.

Para entender los estilos CSS lo más adecuado es pensar en cualquier elemento (<h1>, <p>, <div>, etc.) como una caja. Para cada caja las dimensiones pueden ser controladas para producir una gran variedad de efectos.

Con esta definición, una página es una caja de cajas, y a esta simplificación se le llama modelo de cajas.

Este esquema es básico para conocer cómo afecta un determinado valor a un contenido. Es importante destacar, que en principio este esquema es seguido por todos los navegadores. Sin embargo, algunos navegadores como Internet Explorer lo interpretan de diferente manera.

En las siguientes explicaciones no haremos excepciones y se explicarán los atributos según indica el estándar W3C.



7.1. Unidades de medida

Muchas de las propiedades CSS permiten indicar medidas y colores en sus valores. Además permite indicarlos de formas diferentes. Unidades de medida:

- Para altura, anchura, márgenes, tamaño letra, etc.
- Valores numéricos enteros o decimales seguidos de la unidad de medida.
- Tipos:
 - Absolutas: establecen el valor de forma completa. Valor real es el indicado.
 - Definen las medidas de forma completa. Sus valores reales no se calculan a partir de otro valor de referencia, sino que son los indicados.
 - in (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 las unidades de medida absolutas, la única que se utiliza con frecuencia es la de puntos (pt), para el tamaño de letra.
 - Relativas: definen el valor en relación a otra medida.
 - em: relativa al tamaño de letra empleado. Aunque no es exacto, el valor 1em se aproxima a la anchura de la letra "M" del tipo y tamaño de letra utilizado. Tamaño en puntos. 1em = 12 puntos.
 - ex: relativo a la altura de la letra "x" del tipo y tamaño de la letra utilizada. 1ex = 0.5em
 - px: relativa a la resolución de la pantalla del usuario.Ejemplo:

`body{font-size: 0.9em; } /* El tamaño de la letra de la página será el 90% del tamaño por defecto (que es el tamaño de referencia)*/
body{font-size: .9em; }`
 - Porcentajes: Es otra medida relativa, ya que referencia a otra medida.

Recomendaciones:

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. Se recomienda utilizar la medida "em" y porcentajes para tamaños de letras y pixel y porcentajes para definir el layout (estructura) del documento: anchura de columnas y elementos.

Actividad:

Compruebe en un navegador las diferencias de tamaño con los siguientes códigos. Observar que en este caso el estilo no está definido en el <head> del documento, sino que incorporado en la propia etiqueta (atributo style). Este método viene bien para poner ejemplos rápidos como en este caso, pero no para diseñar debido a su poca escalabilidad.

```
<body>  
<div style="font-size:0.21in">Texto de 0.2 pulgadas </div>  
<div style="font-size:0.3in">Texto de 0.3 pulgadas </div>  
<div style="font-size:0.4in">Texto de 0.4 pulgadas </div>  
<div style="font-size:0.5in">Texto de 0.5 pulgadas </div> </br>  
<div style="font-size:10px">Texto de 10 pixeles </div>  
<div style="font-size:12px">Texto de 12 pixeles </div>  
<div style="font-size:14px">Texto de 14 pixeles </div>  
<div style="font-size:16px">Texto de 16 pixeles </div>  
<div style="font-size:18px">Texto de 18 pixeles </div>  
<div style="font-size:20px">Texto de 20 pixeles </div> </br>  
<div style="font-size:8mm">Texto de 8 milímetros </div>  
<div style="font-size:9mm">Texto de 9 milímetros </div>  
<div style="font-size:10mm">Texto de 10 milímetros </div> </br>  
<div style="font-size:1pc">Texto de 1 pica </div>  
<div style="font-size:2pc">Texto de 2 picas </div>  
<div style="font-size:3pc">Texto de 3 picas </div>  
<div style="font-size:4pc">Texto de 4 picas </div>  
</body>
```

7.2. Atributos de posición

Los atributos de posición son principalmente top, left, right y bottom.

Top (desde arriba) y bottom (desde abajo) indican la distancia en vertical donde se colocará la capa y Left (desde la izquierda) y right (desde la derecha) la horizontal.

Sin embargo, esta distancia está supeditada al atributo position que define el tipo de posición de la capa.

Si el atributo position es absolute (o fixed), top indica la distancia del borde superior de la capa con respecto al borde superior de la página. Si el atributo position era relative, top indica la distancia desde donde se estaba escribiendo en ese momento en la página hasta el borde superior de la capa.

7.3. Atributos margin

Los atributos margin-left, margin-right, margin-top, margin-bottom marcan la separación entre otra caja y el borde de la que se representa. El siguiente ejemplo muestra estos atributos definidos en dos cajas: una para <body> y otra para <div>.

El ejemplo se hace para las dos porque de esa manera se puede ver que las medidas se hacen relativas a la caja que la contiene, en este caso las medidas de los márgenes del <div> se hacen relativas al <body>. Para visualizar mejor el efecto se ha incluido un atributo border que, como veremos más adelante, dibuja el borde de 3 px en rojo para el <div> y azul para el <body>

```
<html>
<head>
<style>
body{
margin: 100px;
margin-right: 100px;
margin-left: 100px;
border: 3px dotted blue;
}
div {
margin-top: 15px;
margin-right : 15px;
margin-bottom : 15px;
margin-left : 15px;
border: 3px dotted red;
}
</style>
</head>
<body>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
</body>
</html>
```

Al no haber relleno (padding) la distancia entre el borde del cuadro exterior (azul) y el más interno (rojo) es de unos 15 px.

La distancia de 100 px es entre el borde exterior (azul) y los bordes de la vista del navegador. Si se incrementan los márgenes de 15 px a 150 px se puede observar que la distancia es mayor entre ambos bordes. También se puede poner una distancia de -15 px en vez de 15 px para conseguir que el borde interior (rojo) salga quede por fuera 15 px del exterior (azul). Sin embargo, no es muy aconsejable usar medidas negativas, para las distancias.

Para simplificar también se puede usar un atributo margin que tiene valores separados por espacios en blanco y cuyo orden es el siguiente (a favor de las agujas del reloj): el primer margen superior(margin-top), el derecho (margin-right), el inferior (margin-bottom) y el izquierdo (margin-left). Si se proporcionan solo dos o tres, los que faltan se asignan automáticamente según la relación con otras cajas. Para el ejemplo anterior, body se podía haber definido de esta manera:

```
body {
margin: 100px 100px 100px 100px;
border 3px dotted blue ; }
```

Los valores de estos atributos (y de cualquier de los siguientes que veremos) también pueden ser auto o inherit.

Calcula automáticamente la mínimo distancia según la relación con otros elementos. Esto es útil cuando las relaciones

se hacen con medidas relativas. Por su lado, inherit hereda el valor del mismo atributo en la caja que lo contiene, es decir, hereda los valores del padre.

7.4. Atributos padding

Padding se puede traducir como relleno. Estos atributos indican la distancia entre el borde y los elementos que se encuentran en el interior. Dicho de otro modo, tienen una función opuesta a la vista anteriormente para el atributo margin. Las medidas que se usan son las mismas que para margin y los nombres son muy parecidos: padding-top (relleno superior), padding-right (relleno derecho), padding-bottom (relleno inferior) y padding-left (relleno izquierdo). El siguiente ejemplo asigna padding-left de 10 px para apreciar la diferencia.

```
<head>
<style>
body {
margin: 100px 100px 100px 100px ;
border: 3px dotted blue ; }
div {
margin-top:15px;
margin-right: 15px;
margin-bottom: 15px;
margin-left: 15px;
padding-left: 10px;
border : 3px dotted red ; }
</style>
</head>
<body>
<div style = "font-size: 0.5in;"> Texto de 0.5 pulgadas </div>
</body>
```

Como ocurre con el atributo margin, también hay un atributo padding que simplifica el escribir el nombre de los 4 atributos. Un ejemplo de uso es el siguiente:

```
body {
padding: 100px 100px 100px 100px;
border : 3px dotted blue;
}
```

7.5. Atributos border

Border-top, border-bottom, border-right, border-left, definen el estilo y color del borde de la caja. Algunas de las palabras clave que tienen son: none, dotted, dashed, solid, double, groove, ridge, inset y outset.

Se puede separar la asignación de color y estilo usando dos atributos separados: border-color y border-style. Estos atributos aplican respectivamente un color y un estilo a todos los bordes de la caja.

También se puede especificar un color y estilo para cada uno de los bordes (top, left, right, bottom) por separado con border-top, border-bottom, border-right, border-left.

Los atributos mostrados no son todos los que definen CSS (versión 3) para manejar bordes. Existen muchos más como por ejemplo, border-radius que se usa para hacer bordes redondeados. Este atributo se utiliza en el siguiente ejemplo:

Por último, también se puede definir por separado el ancho de un borde con la propiedad border-width para todos los bordes de la caja o con border-top-width, border-bottom-width, border-right-width, border-left-width para cada uno por separado.

Los anchos (width) pueden tener como valor un tamaño en cualquier unidad como los vistos anteriormente, o valores predefinidos: thin (estrecho), medium (mediano) o thick (ancho).

El siguiente ejemplo muestra el uso de muchos de los atributos vistos para hacer una caja dentro de otra con borde redondeado.

```
<html>
<head>
<style>
body{
margin: 100px;
border-style:inset;
border-color:blue; /* color del borde azul */
border-radius: 15px; /*borde redondeado de radio 15px */
border-width:thick; /* Un borde grueso */
padding: 15px;
background-color: darkseagreen;
background-size: cover; }
div {
margin: 15px;
padding:10px;
border-top : 3px dotted red ; /*estilo, tamaño y color incluidos en el mismo atributo */
border-right : 2px solid blue ;
border-bottom : 3px double green;
border-left : 3px groove red ;
background-color: chartreuse;
width: 400px;
height: 200px;
}
div:hover{
border-style: groove; }
/* Lista de estilos para bordes:
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
https://www.w3schools.com/css/css\_border.asp */
</style>
</head>
<body>
<div style="font-size:0.5in;">Texto de 0.5 pulgadas</div>
</body>
</html>
```

7.6. Atributos del contenido

También se puede concretar el ancho y alto de un contenido con el atributo `width` y `height`.

`Width` establece la distancia entre el límite del `padding-left` y el `padding-right` (así es como viene definido en el CSS).

`Height` igual pero entre el `padding-bottom` y el `padding-top`.

En esta a sección se ha pretendido hacer una introducción a los atributos más importantes que participan en el modelo de cajas, con la idea de que el desarrollador se forme un mapa mental de cómo pueden estar distribuidos los elementos en una web. Sin embargo, no se han incluido todas las posibilidades que ofrece CSS en este tipo de atributos.

7.7. Actividad 2.6

El siguiente [código](#) muestra una estructura básica de página web con un encabezado, una parte para el menú y otra parte para los contenidos. Este es el esqueleto básico de una web para una Restaurante. Como se puede observar, esta estructura está hecha con clases e identificador con el fin de dar mayor flexibilidad.

Analice el código. Se han definido dos clases asociadas al body: `body.main-sidebar` y `body.sidebar-main`.

Dependiendo de qué clase se use en la etiqueta `<body>` el div asociado a la barra lateral (`sidebar`) se pondrá a la derecha o Izquierda. Por lo tanto, se puede apreciar cómo de fácil es cambiar una distribución con solo cambiar `<body class="main-sidebar">` por `<body class="sidebar-main">`.

Analice el atributo `float`. El valor de `float:left` indica que la caja con este estilo se colocan a la Izquierda y el resto de cajas que se crean posteriores en el código HTML, y que están en la misma posición, se colocan a la derecha de ésta. Esto es lo que le da el efecto de una caja `main` a la izquierda y el `sidebar` apoyada a su derecha. Cuando es `float:right` lo que se indica es que esa caja se coloca a la derecha. Esto da el efecto de una caja `main` a la derecha y el `sidebar` apoyada a su izquierda.

El efecto de un `float` sigue hasta que se usa en otra caja un estilo `clear`.

El problema surge si posteriormente se desea colocar una caja que ya no se apoye a la derecha (o Izquierda) de la flotante sino que se coloque debajo como se hace por defecto (sin atributo `position`). En este caso, esa nueva caja se tiene que crear con un el estilo `clear`. Si se pone `clear:left` se indica que ya no se desea colocar esa caja adyacente a una definida con `float:left`.

Si pone `clear:right`, se indica que ya no se quiere colocar esa caja adyacente a una definida con `float:right`. El estilo `clear` elimina el efecto del `float`.

Modifique este código para conseguir los siguientes efectos:

1. Que el borde de la cabecera sea en azul, solido, redondeado con un ancho de 3 px.
2. Que el borde del menú principal sea de 1 px, en verde y punteado (`dotted`) y con la palabra "Menú Principal" separada 15 px del borde de la caja (solo el borde Izquierdo).
3. ¿Qué ocurre si el borde es de tamaño 5 px?
4. Defina un pie de página igual de ancho y alto que el encabezado pero de color de borde verde (PISTA: Usar en ese nueva caja un estilo con `clear`).
5. ¿Qué ocurre si no se usa un estilo `clear` en el pie de página?"

8. Elementos

Una vez vistos los atributos asociados a las clases. En esta sección se muestran atributos adicionales de carácter más general y relacionados con la apariencia de textos, listas, tablas, enlaces e imágenes.

8.1. Atributos de fuentes

Las propiedades relacionadas con el tipo de letra (fuente) definidas en CSS son:

Font

Permite definir simultáneamente las propiedades relacionadas con el tipo de letra: font-style, font-variant, font-weight, font-size , line-height y font-family (en este orden).

Obligatorio: font-size y font-family. Si se escribe la propiedad line-height debe aparecer separada de font-size por una barra (/).

Otros valores que admite: caption, icon, menu, message-box, small-caption y status-bar (hacen referencia a los tipos de letra que utiliza el sistema operativo para esos elementos.)

Font-family

Permite establecer el tipo de letra (fuente) del elemento.

Se puede indicar la fuente concreta o bien uno de los nombres genéricos: serif, sans-serif, cursive, fantasy o monospace (los navegadores tienen asociada una fuente a cada nombre genérico). Ejemplo:

```
p{
font-family: "Tahoma", "Geneva", san-serif;
}
```

Font-size

Permite establecer el tamaño del tipo de letra (fuente) del elemento.

Valores: tamaño absoluto, tamaño relativo, distancia o porcentaje, aunque generalmente se aconseja utilizar unidades relativas (% o em).

Tamaño absoluto: xx-small, x-small, small, medium, large, x-large y xx-large

Tamaño relativo: larger o smaller.

Valor numérico con unidades absolutas: 16 pt;

Valor numérico con unidades relativas: 150%;

Font-style

Variante letra: inclinada

Valores: normal, oblique o italic.

Font-variant

Letra versalitas (mayúsculas)

Valores: normal y small-caps

Font-weight

Permite elegir el grosor del trazo.

Valores:

- numéricos (100, 200, 300, 400, 500, 600, 700, 800, 900), del más fino al más grueso.
- Normal (valor 400)
- Bold (valor 700)
- Lighter (valor inferior en la lista de valores numéricos)
- Bolder (valor superior en la lista de valores numéricos)

Font-size-adjust

Permite establecer la proporción entre el tamaño del tipo y el de la letra x, indicándolo como número decimal.

Font-stretch

Permite elegir que un tipo de letra más o menos condensado o expandido.

Valores: ultra-condensed, extra-condensed, condensed, semi-condensed, normal, semi-expanded, expanded, extra-expanded, ultra-expanded.

Fuentes de iconos

Las fuentes de iconos son un tipo de fuentes (tipos de letra) en los que cada carácter es un icono que se han hecho populares en los últimos años.

Si a una etiqueta se le asocia la fuente que contiene los iconos, se muestran los iconos correspondientes. Con códigos Unicode.

Las fuentes de iconos se distribuyen con hojas de estilo que contienen todas las clases correspondientes a todos los caracteres para que sólo haya que enlazar la hoja de estilo para poder utilizar los caracteres. Ejemplo:

```
<head>
...
<link rel="stylesheet" type="text/css" href="css/font-awesome.css"/>
<link rel="stylesheet" type="text/css" href="css/estilo.css"/>
</head>
<body>
<p><i class="fa fa-check-circle"></i> Me gusta</p>
<p><i class="fa fa-times-circle"></i>No me gusta</p>
</body>
```

Recopilación de sitios web con iconos:

<https://css-tricks.com/flat-icons-icon-fonts/>

Font Awesome:

<http://fontawesome.io/>

Weather icons:

<http://erikflowers.github.io/weather-icons/>

Explicación sobre el uso de fuentes de iconos:

<http://gomakethings.com/icon-fonts/>

Otras webs de iconos:

IconMonster: <http://iconmonstr.com/>

freepik: <http://www.freepik.com/>

IconFinder: <https://www.iconfinder.com>

IconArchive: <http://www.iconarchive.com/commercialfree.html>

La regla @font-face

Permite utilizar fuentes que no están instaladas en el equipo local.

Funciona a partir de CSS3

Formatos: TTF, OTF, WOFF, WOFF 2.0, SVG Fonts, EOT

Ejemplo:

```
@font-face{
font-family:myFirstFont;
src:url(sensation_light.woff)
}
div{
font-family:myFirstFont;
}
/*ejemplo funcional */
```

```
@font-face {
  font-family: 'Raleway';
  font-style: normal;
  font-weight: 400;
  src: local('Raleway'), local('Raleway-Regular'), url(https://fonts.gstatic.com/s/raleway/v11/0dTEPzKLWceF7z0koJaX1A.woff2) format('woff2');
  unicode-range: U+0000-00FF, U+0131, U+0152-0153, U+02C6, U+02DA, U+02DC, U+2000-206F, U+2074, U+20AC, U+2212, U+2215, U+E0FF, U+EFFD, U+F000;
}
```

```
body {
  font-family: 'Raleway';
  font-style: normal;
  font-weight: 400;
}
```

Google Fonts

Google ofrece un servicio de alojamiento de fuentes libres. Google Fonts.

Google Fonts: las fuentes pueden utilizarse en nuestras páginas web sin necesidad de alojarlas en nuestro propio servidor. También se pueden descargar.

2 formas de incluirlas en nuestras páginas:

- ```
@import url(https://fonts.googleapis.com/css?family=Mystery+Quest);
```

```
p{
font-family: "Mystery Quest", sans-serif;
font-size:150%;
}
```

- ```
<head>
<link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Mystery+Quest" />
</head>
```

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

<link href="https://fonts.googleapis.com/css2?family=Festive&family=Rampart+One&display=swap" rel="stylesheet">

<style>

@import url(https://fonts.googleapis.com/css2?family=Mystery+Quest);

@import url(https://fonts.googleapis.com/css2?family=Festive&display=swap);

.mystery{

font-family: "Mystery Quest", sans-serif;

font-size:150%;

}

.festive{

font-family: "Festive", sans-serif;

}

.rampart{

font-family: 'Rampart One', cursive;

}

</style>

</head>

<body>

<p class="mystery">Fonte de google docs Mystery Quest</p>

<h1 class="festive">Fonte festive</h1>

<h1 class="rampart">Fonte rampart?</h1>

</body>
```

8.2. Propiedades para textos

Propiedad CSS	Valores posibles	Significado
text-align	Left, right, center y justify	Alineación horizontal
vertical-align	baseline, sub, super, top, text-top, middle, bottom y text-bottom	Alineación vertical
text-decoration	none (ninguno), underline,overline (sobrerayado) y line-through (tachado)	Decoración del texto
text-transform	none, capitalize, uppercase y lowercase	Mayúsculas / minúsculas
text-shadow	Color dist_hor dist_ver diámetro	Sombreado del texto
text-overflow	clip ellipsis texto	Comportamiento cuando el texto no cabe
letter-spacing	normal tamaño	Espacio entre letras (tracking)
line-height	normal tamaño	Espacio entre líneas (interlineado)
text-indent	tamaño	Indentación de texto (sangría)
word-spacing	normal tamaño	Espacio entre palabras
white-space	normal nowrap pre pre-line prewrap	Comportamiento de los espacios
tab-size	número de espacios tamaño	Ancho de las tabulaciones
direction	ltr rtl	Dirección del texto
color	nombre color valor RGB	Color del texto

Text-shadow:

```
h1 {
text-shadow: 2px 2px red;
}
```

Texto con sombreado

```
h1{
color: white;
text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;
}
```

Texto con sombra en color azul.

8.3. Tablas

border-collapse

Permite elegir el modo de presentación de los bordes de las celdas y de la tabla.
Valores: initial | collapse | separate | inherit
Valor por defecto: separate

A	B	C	D	E
a	1	2	3	4
b	1	2	3	4
c	1	2	3	4
d	1	2	3	4

Borde por defecto

A	B	C	D	E
a	1	2	3	4
b	1	2	3	4
c	1	2	3	4
d	1	2	3	4

Border-collapse: collapse

border-spacing

Permite establecer una separación entre las celdas cuando se utiliza el modo de bordes separado.
Valores: unidad de medida | inherit
Ejemplo:

These cells	have 5px of spacing	all around them
-------------	---------------------	-----------------

These cells	have 5px of horizontal spacing	and 10px vertical
-------------	--------------------------------	-------------------

```
.horizontal{
border-collapse: separate;
border-spacing: 5px;
border: 1px solid #000;
}
.both{
border-collapse: separate;
border-spacing: 5px 10px;
border: 1px solid #000;
}
```

caption-side

Permite elegir la posición de la leyenda (<caption>) con respecto a la tabla.
Valores: top (arriba), right (derecha), bottom (abajo) y left (izquierda)
• Ejemplos:

<pre>caption { }</pre>	Esto es la leyenda Celda 1Celda 2 Celda 3Celda 4
<pre>caption { caption-side: top; }</pre>	Esto es la leyenda Celda 1Celda 2 Celda 3Celda 4
<pre>caption { caption-side: bottom; }</pre>	Celda 1Celda 2 Celda 3Celda 4 Esto es la leyenda
<pre>caption { caption-side: left; }</pre>	Esto es la Celda 1Celda 2 leyendaCelda 3Celda 4
<pre>caption { caption-side: right; }</pre>	Celda 1Celda 2Esto es Celda 3Celda 4la leyenda

table-layout

Permite elegir el modo en el que el navegador calcula el tamaño total de la tabla y el de cada fila o columna.
Valores: auto | fixed | initial | inherit
En el modo fijo (fixed), el tamaño de las tablas y de las celdas depende de las propiedades width y height.
En el modo fijo (auto), el tamaño de las tablas y de las celdas depende de su contenido (y si es posible, de las propiedades width y height).

Empty-cells

Permite establecer si se muestran o no los bordes de las celdas vacías, en el modo de bordes separado.
Valores: show | hide | initial | inherit
Si no se establece la propiedad, los navegadores aplican el valor show.

<pre>table { }</pre>	Esto es la leyenda Celda 1 Celda 4
<pre>table { empty-cells: show; }</pre>	Esto es la leyenda Celda 1 Celda 4
<pre>table { empty-cells: hide; }</pre>	Esto es la leyenda Celda 1 Celda 4

8.4. Colores y fondos

Colores

Propiedades

color: cambia el color del texto de un elemento.

background-color: Cambia el color de fondo de un elemento.

Los colores en CSS se pueden indicar de cinco formas diferentes:

1. Palabras clave
2. RGB decimal
3. RGB Porcentual
4. RGB Hexadecimal
5. RGBA (CSS3)
6. HSL
7. HSLA

Palabras clave

En la recomendación HTML 3.2 (1997) se incluyeron dieciséis nombres de colores: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white y yellow.

Estos colores son los colores de la paleta VGA de Windows. Casi no se utilizan en CSS. Muy limitados:

Color	Nombre	Código RGB	Color	Nombre	Código RGB
	blue	#0000FF		navy	#000080
	fuchsia	#FF00FF		purple	#800080
	red	#FF0000		maroon	#800000
	yellow	#FFFF00		olive	#808000
	lime	#00FF00		green	#008000
	aqua	#00FFFF		teal	#008080
	gray	#808080		black	#000000
	white	#FFFFFF		silver	#C0C0C0



Además los navegadores modernos soportan muchos otros nombres: [Colores](#)

RGB decimal

RGB: define un color indicando la cantidad de color rojo, verde y azul para obtener el color.

En CSS los componentes de los colores RGB decimal pueden tomar los valores entre 0 y 255.

Ejemplo:

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

RGB Porcentual

Muy similar al RGB decimal. El valor de los componentes RGB puede tomar valores entre 0% y 100%.

Ejemplo:

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

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

RGB Hexadecimal

Es el método más utilizado de indicar los colores. Casi todos los sitios web reales utilizan este método.

Determinar las componentes RGB decimales del color original, por ejemplo: R = 71, G = 98, B = 176.

Transformar a hexadecimal cada componente. En el ejemplo anterior: R=47, G=62, B=B0

El color se indica con # y a continuación cada uno de los componentes: #4762B0

Ejemplo: p {color: #4762B0; }

Se pueden comprimir sus valores cuando son iguales dos a dos:

```
body{background-color: #FFF; color: #000; }  
h1, h2, h3, h4, h5, h6 {color: #C00; }
```

Colores RGBA (CSS3)

Incorporan la transparencia alpha a los códigos RGB mediante un cuarto valor que indica la transparencia del elemento.

La transparencia se expresa como un número decimal entre 0 y 1, en el que el 0 significa completamente transparente y el 1 completamente opaco.

Los códigos RGBA se pueden expresar de distintas formas:

rgba(rojo, verde, azul, transparencia)

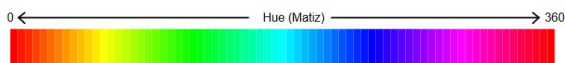
Colores: entre 0 y 255 o bien en porcentaje (0%-100%)

Transparencia: entre 0 y 1

Colores HSL (CSS3)

El color se define mediante tres valores:

Hue (Matiz) es un entero entre 0 y 360 y recorre todos los colores



Saturation (Saturación) es un porcentaje que define la intensidad del color.

Lightness (Luminosidad) es un porcentaje que indica la claridad u oscuridad del color.

Ejemplo:

```
p {  
  background-color: hsl(153, 80%, 40%)  
}
```

Si algo puede ir mal, irá mal.

HSLA (CSS3)

Incorporan la transparencia alpha a los códigos HSL mediante un cuarto valor que indica la transparencia del elemento.

La transparencia se expresa como en el caso de los códigos RGBA con un número decimal entre 0 y 1, en el que el 0 significa completamente transparente y el 1 completamente opaco.

Colores web: <http://hslpicker.com/>

<https://lenguajecss.com/css/colores-y-fondos/colores-css/>

Fondos

- Puede ser un color simple o una imagen.
- El fondo solamente se visualiza en el área ocupada por el contenido y su relleno, ya que el color de los bordes se controla directamente desde los bordes y las zonas de los márgenes siempre son transparentes.
- Para establecer un color o imagen de fondo en la página entera, se debe establecer un fondo al elemento <body>.
- Propiedades:

• **background:** propiedad compuesta para definir fondos.

```
div {  
  background: rgba(0, 128, 0, 0.3) /* Green background with 30% opacity */  
}
```

• **background-color:** establece el color de fondo de un elemento.

• **background-image:** establece cualquier imagen como fondo de un elemento.

```
background-image: url("gradient_bg.png");
```

• **background-repeat:** Por omisión, los navegadores repiten la imagen de fondo tanto en vertical como en horizontal. La propiedad background-repeat permite controlar esa repetición. Valores:

- **no-repeat:** la imagen no se repite.
 - **repeat-x:** la imagen se repite únicamente en horizontal.
 - **repeat-y:** la imagen se repite únicamente en vertical.
 - **repeat:** la imagen se repite en horizontal y vertical.
 - **space:** la imagen se repite en horizontal y vertical, pero se añaden espacios entre las imágenes para mostrar las imágenes completas y que no se corten en los bordes derecho o inferior.
 - **round:** la imagen se repite en horizontal y vertical, pero las imágenes se deforman para mostrarlas completas y que no se corten en los bordes derecho e inferior.
- **background-position:** permite establecer la posición de la imagen de fondo. A esta propiedad hay que darle dos valores: la posición horizontal y la posición vertical. Valores:

- valores numéricos (porcentajes o distancias)
- left, center y right: establecen la posición horizontal (izquierda, centro y derecha, respectivamente)
- top, center, bottom: establece la posición vertical (arriba, en medio, abajo, respectivamente)

- **background-attachment:** establece el comportamiento de la imagen de fondo cuando se desplaza el elemento (al utilizar las barras de desplazamiento del navegador). Valores:
 - scroll: la imagen acompaña al elemento cuando este se desplaza.
 - fixed: la imagen permanece fija
- **background-size:** permite establecer el tamaño de la imagen de fondo. Valores:
 - auto, contain, cover, valores numéricos.
- **background-clip:** permite definir la zona en la que aparece el fondo (color o imagen). Valores:
 - border-box, padding-box, content-box
- **background-origin:** permite definir la zona en la que se sitúa el origen de la posición del fondo. Valores:
 - border-box, padding-box, content-box

Opacity

Permite definir el grado de transparencia de un elemento. Valores:

- <valor_alpha> | inherit
- Valor_alpha: entre 0 y 1
- Valor por defecto: 1

Gradientes

Efecto en el que un color inicial que se transforma poco a poco en uno o varios colores sucesivamente.
Se usa en los fondos.

Gradientes lineales: permite crear fondos con los colores gradientes indicados y una cierta dirección definida.

```
#grad {
  background: red; /* For browsers that do not support gradients */
  background: -webkit-linear-gradient(red, yellow); /* For Safari 5.1 to 6.0 */
  background: -o-linear-gradient(red, yellow); /* For Opera 11.1 to 12.0 */
  background: -moz-linear-gradient(red, yellow); /* For Firefox 3.6 to 15 */
  background: linear-gradient(red, yellow); /* Standard syntax */
}
```

Gradiente Radial: con forma circular.

```
#grad {
  background: red; /* For browsers that do not support gradients */
  background: -webkit-radial-gradient(red, yellow, green); /* Safari 5.1 to 6.0 */
  background: -o-radial-gradient(red, yellow, green); /* For Opera 11.6 to 12.0 */
  background: -moz-radial-gradient(red, yellow, green); /* For Firefox 3.6 to 15 */
  background: radial-gradient(red, yellow, green); /* Standard syntax */
}
```

Gradiente cónico: Un degradado cónico es un degradado con transiciones de color giradas alrededor de un punto central. Para crear un degradado cónico debe definir al menos dos colores.

```
#grad1 {
  background-image: conic-gradient(red, yellow, green, blue, black);
  border-radius: 50%;
}
#grad2 {
  background-image: repeating-conic-gradient(red 10%, yellow 20%);
  border-radius: 50%;
}
```

8.5. Listas

Las propiedades CSS que permiten modificar el aspecto de los elementos de lista son:

list-style: Permite establecer en una sola propiedad los valores de las propiedades list-style-type, list-style-image y liststyle-position. El orden de las propiedades en este caso no influye. El aconsejado:
list-style: <type> <position> <image>

list-style-type: Permite establecer el tipo de viñeta mostrada para una lista.

- list-style-type : disc | circle | square | none . Viñetas sin orden
- list-style-type: decimal | decimal-leading-zero | lower-roman | upper-roman . Viñetas numéricas
- list-style-type: lower-alpha | upper-alpha | lower-greek Viñetas alfabéticas

Valor por defecto: disc

Cuando no se quiere mostrar viñeta o número: none

<ul style="list-style-type: none">• list-style-type: disc• Elemento• Elemento	<ul style="list-style-type: none">◦ list-style-type: circle◦ Elemento◦ Elemento	<ul style="list-style-type: none">■ list-style-type: square■ Elemento■ Elemento
<ul style="list-style-type: none">i. list-style-type: lower-romanii. Elementoiii. Elemento	<ul style="list-style-type: none">α. list-style-type: lower-greekβ. Elementoγ. Elemento	
<ul style="list-style-type: none">A. list-style-type: upper-latinB. ElementoC. Elemento	<ul style="list-style-type: none">a. list-style-type: lower-latinb. Elementoc. Elemento	<ul style="list-style-type: none">Ա. list-style-type: armenianԲ. ElementoԳ. Elemento
<ul style="list-style-type: none">A. list-style-type: upper-alphaB. ElementoC. Elemento	<ul style="list-style-type: none">01. list-style-type: decimal-leading-zero02. Elemento03. Elemento	

```
<ul style="list-style-type: square">  
  <li>list-style-type: square</li>  
  <li>Elemento</li>  
  <li>Elemento</li>  
</ul>
```

list-style-image: Permite reemplazar las viñetas automáticas por una imagen personalizada.

Valores: url | none | inherit

Valor por defecto: none

● Elemento

● Elemento

● Elemento

● Elemento

◆ Elemento

◆ Elemento

◆ Elemento

◆ Elemento

● Elemento

● Elemento

● Elemento

● Elemento

```
ul.ok { list-style-image: url("imagenes/ok.png"); }  
ul.flecha { list-style-image: url("imagenes/flecha.png"); }  
ul.circulo { list-style-image: url("imagenes/circulo_rojo.png"); }
```

list-style-position: Permite establecer la posición de la viñeta de cada elemento de una lista.

Valores inside | outside | inherit

Valor por defecto: outside

<ul style="list-style-type: none">■ list-style-position: outside■ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in diam. Praesent a justo. Nam odio. Quisque a libero vel massa malesuada scelerisque. Curabitur metus.■ Nunc lobortis tortor. Etiam nec nibh. In tincidunt urna ut erat. Integer velit ante, tempus ut, egestas convallis, euismod in, erat.	<ul style="list-style-type: none">■ list-style-position: inside■ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc in diam. Praesent a justo. Nam odio. Quisque a libero vel massa malesuada scelerisque. Curabitur metus.■ Nunc lobortis tortor. Etiam nec nibh. In tincidunt urna ut erat. Integer velit ante, tempus ut, egestas convallis, euismod in, erat.
---	--

Menú vertical con listas: Una buena técnica para el diseño de sitios web es emplear las listas de elementos para crear los menús de navegación.

```
<ul>  
  <li><a href="#">Elemento 1</a></li>  
  <li><a href="#">Elemento 2</a></li>  
  <li><a href="#">Elemento 3</a></li>  
  <li><a href="#">Elemento 4</a></li>  
  <li><a href="#">Elemento 5</a></li>  
  <li><a href="#">Elemento 6</a></li>  
</ul>
```

=>

Elemento 1

Elemento 2

Elemento 3

Elemento 4

Elemento 5

Elemento 6

8.6. Menú vertical con listas

Para crear un menú vertical con listas necesitamos crear el estilo CSS para dicha lista.

Paso 1: definir el ancho del menú

```
ul.menu { width: 180px; }
```

Paso 2: eliminar viñetas, márgenes y espacios

```
ul.menu {  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    width: 180px;  
}
```

Paso 3: Añadir un borde al menú de navegación y establecer el color de fondo y los bordes de cada elemento del menú:

```
ul.menu {  
    border: 1px solid #7C7C7C;  
    border-bottom: none;  
    list-style: none;  
    margin: 0;  
    padding: 0;  
    width: 180px;  
}  
ul.menu li {  
    background: #F4F4F4;  
    border-bottom: 1px solid #7C7C7C;  
    border-top: 1px solid #FFF;  
}
```

Paso 4: Aplicar estilos a los enlaces: mostrarlos como un elemento de bloque para que ocupen todo el espacio de cada del menú, añadir un espacio de relleno y modificar los colores y la decoración por defecto

```
ul.menu li a {  
    color: #333;  
    display: block;  
    padding: .2em 0 .2em .5em;  
    text-decoration: none;  
}
```

8.7. Menú horizontal con listas

Igual que con el menú vertical, debemos crear un estilo para nuestro menú horizontal.

Código HTML del menú horizontal:

```
<ul>
  <li><a href="#">Elemento 1</a></li>
  <li><a href="#">Elemento 2</a></li>
  <li><a href="#">Elemento 3</a></li>
  <li><a href="#">Elemento 4</a></li>
  <li><a href="#">Elemento 5</a></li>
</ul>
```

Paso 1: Aplicar los estilos CSS básicos para establecer el estilo del menú (similares a los del menú vertical anterior):

```
ul.menu {
  background: #84P4F4;
  border: 1px solid #7C7C7C;
  list-style: none;
  margin: 0;
  padding: 0;
}

ul.menu li a {
  color: #333;
  display: block;
  padding: .3em;
  text-decoration: none;
}
```

Paso 2: Establecer la anchura de los elementos del menú. En este caso se asigna un ancho de 20% a cada elemento. Se posiciona de modo flotante.

```
ul.menu li {
  float: left;
  width: 20%;
}
```

```
ul.menu {
  overflow: hidden;
}
```

Paso 3: Establecer los bordes de los elementos que forman el menú.

```
ul.menu li a {
  border-left: 1px solid #FFF;
  border-right: 1px solid #7C7C7C;
}
```

Paso 4: se elimina el borde derecho del último elemento de la lista, para evitar el borde duplicado.

```
<ul>
  <li><a href="#">Elemento 1</a></li>
  <li><a href="#">Elemento 2</a></li>
  <li><a href="#">Elemento 3</a></li>
  <li><a href="#">Elemento 4</a></li>
  <li><a href="#" style="border-right: none">Elemento 5</a></li>
</ul>
```

8.8. Pseudo-elementos

Se refiere a una parte de un elemento que, aunque actúa como éste, no es un elemento en sí.

Ejemplo: la primera letra de un párrafo, o la primera línea....

Hay 5 tipos (En CSS3 se añaden los :: delante para diferenciarlos de los pseudo-elementos de CSS2.1)

:first-letter

Permite seleccionar la primera letra de la primera línea de texto de un elemento.

Ejemplo:

```
p{
  font-size: 14px;
}

p:first-letter {
  font-size: 2em;
}
```

Esta regla muestra la primera letra de cada párrafo con el doble de tamaño.

:first-line

Permite seleccionar la primera línea de texto de un elemento.

Ejemplo:

```
p:first-line {
  text-transform: uppercase;
}
```

Esta regla muestra en mayúscula la primera línea de cada párrafo.

:first-child

Permite seleccionar el primer elemento hijo de un elemento.

Ejemplo:

```
div p:first-child {
  color: red;
}
```

Esta regla muestra como identificar al primer párrafo dentro de un bloque <div> sin necesidad de asignar ninguna clase al párrafo.

:before

Permite insertar contenido antes de un elemento.

Ejemplo:

```
1 p.errores:before {
2   content: "ERRORES COMUNES";
3 }
1 | <p class="errores">Contenido del párrafo..</p>
```

Esta regla permite que cuando encuentre un párrafo de clase errores, inserte justo antes del mismo "ERRORES COMUNES".

:after

Hace justo lo contrario a :before, es decir, permite insertar contenido después de un elemento.

Ejemplo:

```
1 p.ejemplo:after {
2   content: "";
3   display: block;
4   width: 200px;
5   height: 50px;
6   border:1px solid red;
7 }
```

Siempre tienen que llevar el atributo content, sino no funcionarán.

8.9. Enlaces

Con las pseudo-classes es posible aplicar efectos especiales a determinados elementos de una página HTML, como a enlaces o imágenes, cuando se realizan algunas acciones específicas con el ratón, como pasar sobre una imagen o enlace, o hacer clic en ellos...

:link

Usamos la pseudo-clase :link para seleccionar todas las etiquetas de enlaces a las que aún no se les ha dado clic.

:visited

Se aplica a todos los enlaces que han sido visitado al menos una vez por el usuario.

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

a:visited {
    color: green;
}
```

:hover

Se aplica a cualquier elemento HTML, no sólo a enlaces. Se activa cuando el usuario pasa el ratón por encima de cualquier elemento.

:active

Se produce cuando el usuario activa un elemento, por ejemplo cuando se pulsa el ratón sobre un elemento. Se aplica durante un espacio de tiempo muy pequeño.

:focus

Permite aplicar un estilo a un elemento cuando éste recibe el foco, es decir, la atención del usuario.

```
<style>
a.menu:link {
    text-decoration: none;
    color: #000000;
    border: #FFFFFF 1px solid;
}
/* Link no visitado*/
a.menu:visited {
    text-decoration: none;
    color: #cccccc;
}
/*Link visitado*/
a.menu:active {
    text-decoration: none;
    color: #003399;
    background: green;
    border: #FFFFFF 1px solid;
}
/*Link activo*/
a.menu:hover {
    text-decoration: underline;
    color: #003399;
    background: red;
    border: #FFFFFF 1px solid;
}
/*Ratón sobre el link*/
.centrado{
    position: absolute;
    left: 0px;
    right: 0px;
    background-color: brown;
    width: 100px;
    height: 100px;
    margin:15px auto 15px auto;
}
</style>

<body>
<a href="#" class="menu">Enlace uno</a>
<a href="#" class="menu">Enlace dos </a>
<a href="#" class="menu">Enlace tres</a>
<div class="centrado">¡hola</div>

</body>
```

Actividad:

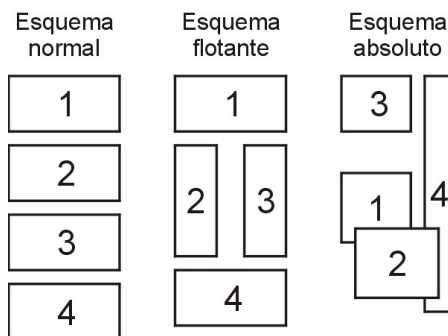
Modifique el código de la Actividad 2.6 para mostrar los siguientes elementos:

- a. Dividir el encabezado header en dos partes. Colocar un logo acorde con la página en una de ellas y un texto <h1> a lado del logo.
- b. Colocar una imagen en el contenido main.
- c. Hacer un menú vertical con 4 enlaces en el sidebar que no lleve a ningún enlace pero que si tenga atributos para sus enlaces: activo, hover, etc.

9. Posicionamiento

Los elementos de una página web están contenidos en una caja rectangular, de acuerdo con el modelo de caja. La manera en que los diferentes elementos de una página web se distribuyen en la pantalla dependen del esquema de posicionamiento elegido. Existen tres esquemas de posicionamiento:

- normal (por defecto)
- flotante (elemento no ocupa todo el ancho, se coloca a la izda o la dcha de otros elementos)
- absoluto (en cualquier posición)



Posicionamiento Flotante:

float

Establece el esquema de posicionamiento flotante para un elemento. Se puede aplicar a cualquier elemento.

Valores: left y right

Esta propiedad no permite centrar -> text-align: center

El tamaño de los elementos flotantes hay que establecerlo con las propiedades width y height.

```
p {
  border: black 2px solid;
  float: left;
  height: 100px;
  margin: 3px;
  padding: 3px;
  width: 100px;
}
```

Esto es el primer párrafo	Esto es el segundo párrafo	Esto es el tercer párrafo	Esto es el cuarto párrafo	Esto es el quinto párrafo	Esto es el sexto párrafo
---------------------------	----------------------------	---------------------------	---------------------------	---------------------------	--------------------------

clear

Permite que un elemento no tenga elementos flotantes a su lado.

Valores:

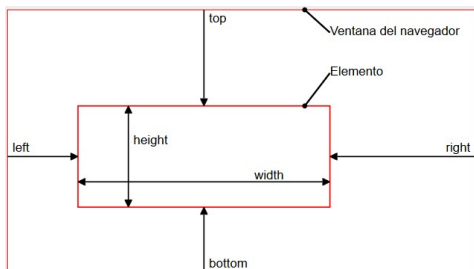
- left, hace que no haya elementos flotantes a la izquierda
- right, hace que no haya elementos flotantes a la derecha
- both, hace que no haya elementos flotantes ni a derecha ni a izquierda
- none, permite que haya elementos flotantes a derecha y a izquierda (valor por omisión).

Posicionamiento absoluto:

Position. Establece el esquema de posicionamiento absoluto de un elemento. Las 4 esquemas de posicionamiento absoluto:

- **Static:** es el valor predeterminado del atributo y el posicionamiento normal de los elementos en la página. Quiere decir que los elementos se colocarán según el flujo normal del HTML, es decir, según estén escritos en el propio código HTML. Por decirlo de otra manera, static no provoca ningún posicionamiento especial de los elementos y por tanto, los atributos top, left, right y bottom no se tendrán en cuenta.
- **Relative:** indica que la capa si forma parte del flujo normal de elementos de la página, por lo que su posición dependerá del lugar donde esté en el código y el flujo HTML. Además, las capas con posicionamiento relative, admiten los valores top y left para definir la distancia a la que se colocan con respecto al punto donde esté en ese momento el flujo normal del HTML. Como afectan al mencionado flujo del HTML, los elementos colocados después de las capas estilo relative, tendrán en cuenta sus dimensiones para continuar el flujo y saber dónde colocarse.
- **Fixed:** este atributo sirve para posicionar una capa con posicionamiento absoluto, pero su posición final será siempre fija, es decir, aunque se desplace el documento con las barras de desplazamiento del navegador, siempre aparecerá en la misma posición. El lugar donde se anclará la capa siempre es relativo al cuerpo (el espacio disponible del navegador para la página). Si utilizamos top y left, estaremos marcando su posición con respecto a la esquina superior izquierda y si utilizamos bottom y right su posición será relativa a la esquina inferior derecha.
- **Absolute:** el valor absolute permite posicionar cajas de manera absoluta, esto es de manera definida por valores de los atributos top, left, bottom y right. Las capas o elementos con posicionamiento absoluto quedan aparte del flujo normal del HTML no viéndose afectadas por el lugar en donde aparezcan dentro del HTML y tampoco afecta estas cajas a otras del flujo normal del HTML.
- **Sticky:** elemento fijo alterna entre relative y fixed, según la posición de desplazamiento. Se coloca en relación hasta que se alcanza una posición de desplazamiento determinada en la ventana gráfica.

Es importante destacar que los valores top, left, bottom y right son una distancia con respecto al primer elemento contenedor que tenga un valor de position distinto de static. Si todos los contenedores donde esté la capa posicionada con estilo absolute (todos sus padres hasta llegar a <body> son static, simplemente se posiciona con respecto al lado superior de la página, para el caso de top, el inferior para bottom, del lado izquierdo para left o el derecho, en el caso de utilizar right.



- top: posición del lado superior del elemento contando desde el borde superior de la ventana (de arriba a abajo, es decir, cuanto más grande es el valor más hacia abajo se coloca el elemento).
- bottom: posición del lado inferior del elemento contando desde el borde inferior de la ventana (de abajo a arriba, es decir, cuanto más grande es el valor más hacia arriba se coloca el elemento).
- height: altura del elemento.
- left: posición del lado izquierdo del elemento contando desde el borde izquierdo de la ventana (de izquierda a derecha, es decir, cuanto más grande es el valor más hacia la derecha se coloca el elemento).
- right: posición del lado derecho del elemento contando desde el borde derecho de la ventana (de izquierda a derecha, es decir, cuanto más grande es el valor más hacia la izquierda se coloca el elemento).
- width: anchura del elemento.

<https://www.mcilbr.org/consultar/htmlcss/css/css-posicionamiento-absoluto.html>

Prueba el siguiente código:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <meta http-equiv="X-UA-Compatible" content="ie=edge">

  <title>Document</title>

  <style>

    .capal{

      z-index: 9;

      position: absolute;

      background-color: #686;

      color:#ffc;

    }
```

```
padding:10px;

text-align: center;

width:300px;

/*top:0;*/

}

.capa2{

    z-index: 1;

    position: relative;

    width: 300px;

    padding: 10px;

    background-color: #066;

    color:#ffc;

    left: 30px;

}

.capa3{

    z-index: 1;

    position: fixed;

    bottom: 0px;

    width: 300px;

    padding: 10px;

    background-color: #066;

    color:#ffc;

    left: 30px;

}

.capa4{

    z-index: 19;

    position: sticky;

    width: 300px;

    padding: 10px;

    background-color:crimson;

    color:#ffc;

    left: 30px;

    top:0px;

    bottom: 10px;

}

</style>

</head>

<body>

    <h1>No define posición</h1>

    <div class="capa1">No define posición, por lo que es absolute</div>

    <div class="capa2">Capa de posicionamiento relative<br>Se tiene en cuenta esta capa para posicionar las siguientes.</div>

    <h2>la última en ponerse</h2>

    <br><br><br><br><br><br><br><br><br><br><br><br><br>

    <br><br><br><br><br><br><br><br><br><br><br><br><br>

    <h1>Fondo de la página</h1>

    <br>

    <div class="capa4">Capa 4 sticky</div>

    <br><br><br><br><br><br>

    <br><br><br><br><br><br>

    <br><br><br><br><br><br>

    <br><br><br><br><br><br>

    <br><br><br><br><br><br>

    <div class="capa3">Pie de página</div>

</body>

</html>

Prueba el siguiente código:

<body>

<h1>NO define posición</h1>

<div style="background-color: #066; color:#ffc; padding:10px; text-align: center; width: 300px;">No define posición, por lo que es static</div>

<div style="position: absolute; width: 300px; padding: 10px; background-color: #066; text-align:right; color:#ffc; top:100px; left: 322px;">Capa Absoluta (solo tiene como contenedor a "body") por lo tanto es relativa a body. </div>

<div style="position: relative; width: 300px; padding: 10px; background-color: #033; color:#ffc; top:0px; left: 30px;">Capa de posicionamiento relative. Se coloca según la posición que le tocará en el flujo HTML.</div>

<div style="position: static; width: 300px; padding: 10px; background-color: #096; color:#ffc; top:10px; left: 30px;">Capa Fija se coloca como una absoluta</div>
```

```
<div style="position: absolute; width: 350px; height:100px; padding: 10px; background-color: #666; text-align:right; color:#ffc; top:300px; left: 500px;">

  <div style="position: absolute; width: 300px; padding: 10px; background-color: #111; text-align:right; color:#ffc; top:0px;

    left: 5px;"> Capa absoluta, se referencia segun a su contenedor porque no es un static </div>

</div>

<h2>La última en ponerse</h2>

</body>
```

10. Visualización

CSS define cuatro propiedades para controlar el modo de visualización de los elementos: **display**, **visibility**, **overflow** y **z-index**

Utilizando algunas de estas propiedades es posible ocultar y/o hacer invisibles las cajas de los elementos, por lo que son importantes para realizar efectos avanzados y animaciones.

display

Permite controlar la forma de visualizar un elemento u ocultar completamente el mismo, haciendo que desaparezca de la página.

Como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Se aplica a cualquier elemento

Valores: inline | block | none | list-item | run-in | inline-block | table | inlinetable | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | tablecaption | inherit

Valor por defecto: inline

Ejemplo de utilización: la propiedad display: inline se puede utilizar en las listas (,) que se quieren mostrar horizontalmente y la propiedad display: block se emplea frecuentemente para los enlaces que forman el menú de navegación.

visibility

Permite hacer visible o invisible un elemento de la página.

Si se hace invisible, el navegador crea la caja del elemento pero no la muestra. En este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

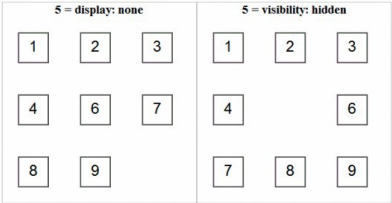
Se aplica a todos los elementos.

Valores: visible | hidden | collapse | inherit

Valor por defecto: visible

El valor collapse de la propiedad visibility sólo se puede utilizar en las filas, grupos de filas, columnas y grupos de columnas de una tabla.

Ejemplo: diferencia entre ocultar una caja (5) mediante las propiedades display o visibility



overflow

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

CSS define la propiedad overflow para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

Se aplica a elementos de bloque y celdas de tablas.

Valores visible | hidden | scroll | auto | inherit

Valor por defecto: visible



z-index

CSS permite controlar la posición tridimensional de las cajas posicionadas. De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

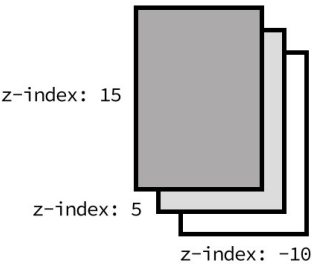
La posición tridimensional de un elemento se establece sobre un tercer eje llamado Z y se controla mediante la propiedad z-index.

Se aplica a elementos con posicionamiento absoluto (mediante la propiedad position).

Valores: auto | número | inherit

Valor inicial: auto

A mayor número, el elemento se posiciona encima.



Flexbox

Es un sistema de elementos flexibles en la que los elementos HTML se adaptan y colocan automáticamente y es más fácil personalizar los diseños. Elementos de este esquema:

- Contenedor
- Item
- Eje principal
- Eje secundario

Para utilizar este modo de colocar los elementos en la página se emplea la propiedad display, con los valores:
inline-flex: Establece un contenedor de ítems flexible en línea, de forma equivalente a inline-block.
flex: Establece un contenedor de ítems flexible en bloque, de forma equivalente a block.
De esta manera los elementos no se desbordarán ni tendrán problemas con los porcentajes.

Dirección de los ejes:
flex-direction: row | row-reverse | column | column-reverse . Cambia la orientación del eje principal.
flex-wrap: nowrap | wrap | wrap-reverse . Evita o permite el desbordamiento (multilínea).

Ejemplo:

```
#contenedor{
background: #CCC;
display: flex;
flex-direction: column;
.item{
background: #777;
}
#contenedor2{
background: #CCC;
display: flex;
width: 200px;
flex-wrap: wrap;
.item2{
background: #777;
}
<div id="contenedor">
<div class="item item-1">1</div>
<div class="item item-1">2</div>
<div class="item item-1">3</div>
</div>
```

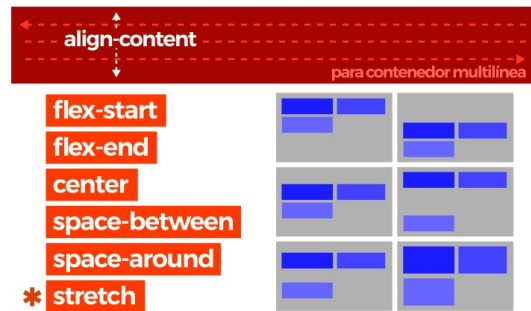
Propiedades de alineación de ítems:

- justify-content:** flex-start | flex-end | center | spacebetween | space-around Eje principal
- align-content:** flex-start | flex-end | center | spacebetween | space-around | stretch Eje principal
- align-items:** flex-start | flex-end | center | stretch | baseline Eje secundario
- align-self:** auto | flex-start | flex-end | center | stretch | baseline Eje secundario

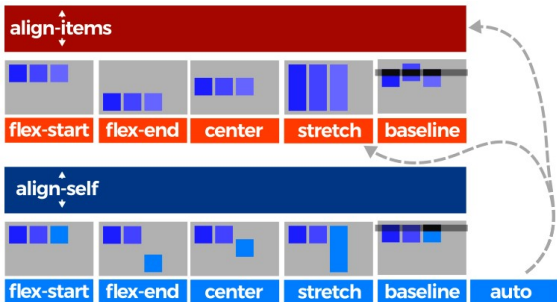
Alineación de ítems en el eje principal:



Alineación de ítems en el eje principal multilínea



Alineación en el eje secundario



- Propiedades que se aplican a los elementos hijos (del contenedor):
- flex-grow:** 0 | [factor de crecimiento] . Número que indica el crecimiento del ítem respecto al resto.
- flex-shrink:** 1 | [factor de decrecimiento] . Número que indica el decrecimiento del ítem respecto al resto.
- flex-basis:** [tamaño base] | content . Tamaño base de los ítems antes de aplicar variación.
- order:** [número] Número que indica el orden de aparición de los ítems.

Cuando usar flexbox:

https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Typical_Use_Cases_of_Flexbox

12. Box-sizing

La propiedad CSS box-sizing se usa para alterar el modelo de caja por defecto usado para calcular alturas y anchuras de elementos.

Valores: content-box | padding-box | border-box

Por defecto: content-box

- content-box: width y height se miden incluyendo sólo el contenido, pero no el borde, margen o relleno.
- padding-box: width y height incluyen el tamaño del relleno pero no incluyen el borden ni margen.
- border-box: width y height incluyen el relleno y el borde, pero no el margen.

Comparando con el modelo de cajas:

```
<div style="width:200px;
padding: 20px;
border: 10px solid #ccc;
margin: 0 auto;">
Lorem ipsum...
</div>
```

El elemento <div> no mide 200px de ancho, sino 260px. Es decir:

200px de ancho inicial, más 20px de padding izquierdo, más 20px de padding derecho, más 10px de borde izquierdo, más 10px de borde derecho.

Equivalente a box-sizing:content-box

box-sizing: border-box

```
<div style="width: 200px;
padding: 20px;
border: 10px solid #ccc;
-webkit-box-sizing: border-box;
-moz-box-siring: border-box;
box-sizing: border-box;
margin: 0 auto;">
Lorem ipsum...
</div>
```

En este caso el elemento <div> ocupa exactamente 200px, ni uno más, ni uno menos.

Esto es muy útil para elementos fluidos, cuando necesitas que el elemento ocupe (por ejemplo) un 33% del ancho, y si ocupa un píxel más toda la estructura se descoloca.

box-sizing: padding-box

```
<div style="width:200px;
padding: 20px;
border: 10px solid #ccc;
-webkit-box-sizing: padding-box;
-moz-box-siring: padding-box;
box-sizing: padding-box;
margin: 0 auto;">
Lorem ipsum...
</div>
```

En este caso el elemento es 220px de ancho (200 entre contenido y padding y 10 de cada lado de borde).

No es tan útil como el caso anterior.

13. Filtros CSS

Permite aplicar ciertos efectos de imagen propios de aplicaciones de retoque fotográfico como sepia o variación de contraste al vuelo, sin hacer cambios permanentes sobre una imagen.

Propiedad: filter Ejemplo de uso:

```
img {  
  filter: grayscale(75%);  
}
```

grayscale: Escala de blanco y negro. Porcentaje. 0% = original, 100% = B&N

blur: Grado de difuminado píxeles. Radio de desenfoque (en píxeles)

sepia: Grado de color sepia porcentaje 0% = original, 100% = sepia

saturate: Grado de saturación porcentaje 0% = B&N, 100% = original, permite >100%

opacity: Grado de transparencia porcentaje 0% = invisible, 100% = visible

brightness: Brillo porcentaje 0 = negro, 100% = original, permite >100%

contrast: Contraste porcentaje 0 = gris, 100% = original, permite >100%

hue-rotate: Rotación de color (matiz) grados 0 - 360deg = original

invert: Invertir porcentaje 0% original, 100% = invertido

drop-shadow: Sombra idéntica x y blur color

<https://devcode.la/tutoriales/filtros-de-instagram-con-css/>
<https://developer.mozilla.org/es/docs/Web/CSS/filter>

13.1. Modos de fusión

Propiedad: **mix-blend-mode**
Permite utilizar los denominados modos de fusión, muy comunes en programas profesionales de retoque fotográfico, sobre elementos de contenido como imágenes o similares.

Propiedad: **background-blend-mode**
Para aplicar el modo de fusión a imágenes de fondo mediante las propiedades background-image o background.

```
body {
  background: green;
}

img {
  mix-blend-mode: multiply;
}
```

Modo de fusión	Significado
normal	Sin modo de fusión.
multiply	Multiplicar.
screen	Trama.
overlay	Superponer.
darken	Oscurecer.
lighten	Aclarar.
color-dodge	Sobrexponer color.
color-burn	Subexponer color.
hard-light	Luz fuerte.
soft-light	Luz suave.
difference	Diferencia.
exclusion	Exclusión.
hue	Tono.
saturation	Saturación.
color	Color.
luminosity	Luminosidad.

<https://css-tricks.com/almanac/properties/m/mix-blend-mode/>

14. Precedencia de estilos

La precedencia de estilos es una manera de indicar que un estilo definido prevalece por encima de otro definido en la misma o en CSS diferentes. Esto es necesario principalmente cuando hay dos o más estilos que actúan sobre los mismos atributos pero con diferente valor.

La precedencia de estilos va asociado con el concepto de especificidad de una regla. La especificidad se refiere al peso que toman cada uno de los elementos de una hoja de estilo. Cuanto más peso más especificidad. Cuanta más especificidad tenga un regla menos problemas a la hora de garantizar que será esa y no otra la regla que se aplique sobre un determinado contenido.

Un cálculo sencillo para calcular la especificidad de una regla es sumar los puntos según el tipo de selectores que contenga:

- Se da un valor de 1 punto a un selector de etiqueta (por ejemplo, <h>, <p>, <div>).
- A un selector de clase se le da el valor de 10 puntos.
- A un selector de identificador se le da un valor de 100 puntos.
- A un atributo de estilo a los que se les da un valor de 1.000 puntos.

Con este cálculo, si se desea agregar un estilo a los párrafos <p> de una página agregando un selector p {estilo}, este será utilizado para dar estilo a todos los párrafos del HTML, pero solo se le dará el valor total de 1 punto. Un punto es muy poco especificidad. Cualquier regla con especificidad> 1 se aplicaría antes.

Si se define una clase .parrafo {estilo} que se aplique sobre los párrafos tendrá una especificidad de 10 puntos. Con lo cual, los párrafos <p> que usen el estilo de la clase .parrafo tendrán como estilo lo de la clase antes que los definidos con p{estilo}. De la misma manera, si en vez de clase se define un selector de identificador #id-parrafo{estilo} esta tomará el valor de 100 puntos de especificidad, por lo que será la más importante que los de clase y el de etiqueta definidos antes. Es importante destacar que, en el caso de que dos o más reglas en conflicto tengan el mismo valor de especificidad, se aplicará la última definida. El siguiente código muestra las especificidades de los selectores definidos:

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <meta http-equiv="X-UA-Compatible" content="ie=edge">

  <title>Document</title>

  <link rel="stylesheet" href="miestilo1.css" type="text/css" media="screen">

</head>

<body>

<p>El fondo de este párrafo será de color carmesi</p>

<p class="parrafo">El fondo de este párrafo será de color granate</p>

<div class="parrafo">

  El fondo de este párrafo será de color rosa

  <p>El fondo de esta línea es carmesi</p>

  El fondo de este párrafo vuelve ser de color rosa

</div>

<p id="id-parrafo" style="background: black; color: white;"> El fondo de este párrafo será negro porque usa un atributo de estilo (con peso 1000) dentro de la etiqueta </p>

<p id="id-parrafo">El fondo de este párrafo será de color rojo</p>

<p id="id-parrafo" class="parrafo">El fondo es de color verde</p>

</body>

</html>
```

Además de la especificidad, se puede obligar a que un atributo de una regla se aplique por encima del resto de reglas. Esto se hace usando la declaración !important. Para utilizar !important en una regla de estilo, siempre se coloca en la parte del valor del atributo, antes del punto y coma",". Por ejemplo:

```
p{
background: red !important;
background: crimson;
}
```

Tenemos una declaración de estilos para los párrafos <p>, donde definimos dos veces el atributo background. En condiciones normales, se tendría en cuenta el valor definido en segundo lugar. Sin embargo, al estar el primer background definido como !important en realidad lo que ocurrirá es que prevalezca este atributo sobre el otro y se aplique un color de fondo rojo sobre todos los párrafos. Si se sustituye esta declaración de p vista en el ejemplo anterior (p{background: crimson;}) por esta otra (p{background: red !important; background: crimson ;}) el resultado será que todos los párrafos tendrán color de fondo rojo (el div seguirá siendo rosa porque no es una párrafo y no se aplica sobre él esta regla).

15. Crear y vincular hojas de estilo

En este punto se describen las alternativas existentes para asociar un código HTML (contenido) con un estilo determinado y definido en CSS. Existen tres alternativas principales, y todas ellas han sido mostradas en los diferentes ejemplos del capítulo. La primera alternativa es usando el atributo `style` dentro de las etiquetas HTML (reglas de estilos integradas).

Por ejemplo:

```
<p style="background: black;">El fondo de este párrafo será negro </p>
```

Esta alternativa presenta alta prioridad tiene frente a otras reglas. Sin embargo esta aparente ventaja se convierte en desventaja cuando, al usarla, se pierde la posibilidad de reutilizar reglas entre elementos, teniendo que escribir todos los atributos de nuevo cuando se desea aplicar el mismo estilo a otros elementos. Además, otra desventaja añadida es que se dificulta el mantenimiento de las CSS haciendo más complicados los cambios y la localización de errores.

Otra alternativa es usando la etiqueta `<style>` dentro del mismo fichero (reglas de estilo incrustadas). En este caso la etiqueta `<style>` se coloca en la cabecera `<head>` del documento. La etiqueta `<style>` tiene varios atributos opcionales:

- **Type** (requerido). se usa para indicar que se aplica un estilo formato CSS.
- **Media**: atributo para indicar sobre qué dispositivo se aplicarán los estilos. Algunos de los valores posibles son: `handheld` (para dispositivos móviles), `print` (para salida por impresora), `projection` (para presentación en proyectores), `screen` (para pantallas), `tv` (para televisores), `braille` (para presentación en dispositivos braille) u `all` (para todos los dispositivos). Si se desean especificar varios valores se puede hacer separados por comas (,).
- **Title**: nombre que se le da al estilo. Útil cuando se vinculan CSS externas.

Un ejemplo de declaración es el siguiente:

```
<style type="text/css" media="screen, tv" title="Mi Estilo 1">
```

En la mayor parte de los ejemplos utilizados en este capítulo se ha sido la etiqueta `<style>` aplicada a un único HTML. Este método tiene sentido cuando un único documento tenga un único estilo. Si la misma hoja de estilo se usa en múltiples documentos o páginas web, entonces sería más apropiado disponer de una hoja de estilo externa tal y como se verá en la siguiente sección.

16. Crear y vincular hojas de estilo en cascada externa

La ventaja de utilizar hojas de estilo externas (CSS externas) es que se pueden reutilizar en varios documentos HTML, permitiendo así, por ejemplo, que todo un sitio web se rija por las misma CSS. De hecho, esta alternativa es la más empleada profesionalmente.

Una hoja de estilo externa puede ser enlazada a un documento HTML mediante la etiqueta <link> que se coloca en el <head> de la página. El siguiente ejemplo muestra cuatro enlaces a hojas CSS.

```
<link rel=stylesheet href="estilo.css" type="text/css" media=screen>
<link rel=stylesheet href="color-8b.css" type="text/css" title="estilo de color 8bit" media="screen, print">
<link rel="alternate stylesheet" href="color-24b.css" type="text/css" title="estilo de color 24-bit" media="screen, print">
<link rel=stylesheet href="aural.css" type="text/css" media=screen>
```

Cuando se define una CSS externa ésta no debe contener ninguna etiqueta HTML como <head> o <style>. La hoja de estilo solo debería consistir de reglas de estilo o sentencias. Un archivo que solo consista de la siguiente línea podría utilizarse como hoja de estilo externa:

```
p { margin: 2em }
```

El atributo rel se usa para definir la relación entre el archivo enlazado y el documento HTML. rel=stylesheet especifica un estilo persistente o preferido. Un estilo persistente es aquel que siempre se aplica si están activas las hojas de estilo. Un estilo preferido es uno que se aplica automáticamente, como en la segunda etiqueta <link> en el ejemplo. La combinación de rel=stylesheet y un atributo title especifica un estilo preferido. Los autores no pueden especificar más de un estilo preferido.

Por otro lado rel="alternate stylesheet" define un estilo alternativo. Un estilo alterno se indica por rel="alternate stylesheet".

La tercera etiqueta <link> en el ejemplo define un estilo alternativo, que el usuario podría elegir para reemplazar la hoja de estilo preferido. Debe tenerse en cuenta que algunos navegadores carecen de la capacidad de elegir estilos alternativos.

Un estilo simple también puede ser dado mediante múltiples hojas de estilo. Todas ellas contribuyen a dar el estilo al HTML que las importa:

```
<link rel=stylesheet href="basico.css" title="miestilo">
<link rel=stylesheet href="tablas.css" title="miestilo">
<link rel=stylesheet href="formas. css" title="miestilo">
```

En este ejemplo, tres hojas de estilo son combinadas en un estilo llamado "miestilo" que se aplica como una hoja de estilo preferido. Para combinar múltiples hojas de estilo en un estilo único, se debe usar el mismo title con cada hoja de estilo.

Como se ha comentado al principio, una hoja de estilo externa es ideal cuando el estilo se aplica a muchas páginas. Un autor podrá cambiar la apariencia de un sitio completo mediante el cambio de un solo archivo. Además, la mayoría de navegadores guardan en caché las hojas de estilo externas, evitando así una demora en la presentación una vez que la hoja de estilo se ha guardado en caché.

Otra alternativa para enlazar CSS externos es usar la regla @import. Esta regla va incluida dentro de las etiquetas <style>.

Un ejemplo de uso de esta regla es el siguiente:

```
<style>@import url("estilos.css");</style>
```

El funcionamiento es igual que usar <link>. La diferencia es que @import no es soportada por todos los navegadores y, además, <link> ofrece mejores alternativas si se desea usar hojas de estilo preferentes, alternativas o preferidas.

Aún así, @import permite hacer ciertas cosas como elegir cuál importar dependiendo del medio (media) al que se vaya a aplicar.

```
<style>
@import url("impresora.css")print;
@import url("normal.css")screen;
</style>
```

17. Diseño web adaptativo

Es una de las características que más se ha popularizado en los últimos años.