



UNIDAD 6. LENGUAJES DE MARCAS EN EL ENTORNO WEB. CSS

1º Desarrollo de Aplicaciones Web
Lenguajes de Marcas y Sistemas de
Gestión de Información

1. ORIGEN DE CSS



- ▶ CSS son las siglas de **Cascade Style Sheet** que traducido significa hojas de estilo en cascada.
- ▶ Las hojas de estilo es una tecnología que nos permite controlar la apariencia de una página web. En un principio, los sitios web se concentraban más en su contenido que en su presentación. HTML no pone mucha atención en la apariencia del documento. CSS describe como los elementos dispuestos en la página son presentados al usuario.
- ▶ Con CSS podemos especificar estilos como el tamaño, fuentes, color, espaciado entre textos y recuadros así como el lugar donde disponer texto e imágenes en la página.

1. ORIGEN DE CSS

- ▶ En un principio en el mismo documento conviven estructuras e instrucciones de formato → resulta un archivo muy “complejo” al estar mezclados.
- ▶ El W3C crea un sistema en el que estructura y formato están separados.
- ▶ Rápidamente se encuentra la utilidad, ya que para cambiar el aspecto de la página sólo se modifican algunas líneas de código → ahorro de tiempo y recursos.
- ▶ CSS evoluciona día a día y ofrece cada vez más posibilidades.

2. SINTAXIS

Una **regla de estilo** sigue el siguiente formato:

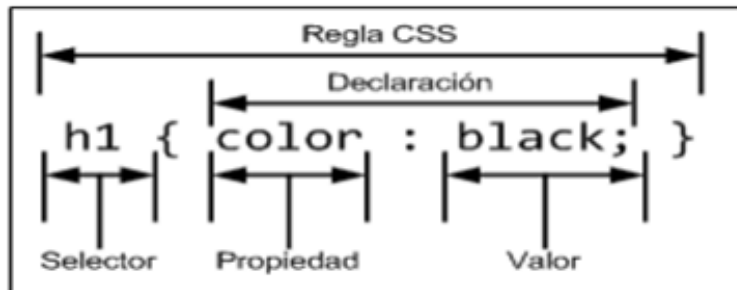
```
p {color: blue;}
```

1 2 3

Selector Propiedad Valor

Con esta regla, los párrafos de la página web tendrán la letra de color azul.

1. Determina qué elemento HTML se ve afectado (p).
2. Indica a qué propiedad del elemento HTML se le va a dar un estilo o formato concreto (color).
3. Establece el valor que se le da a la propiedad del elemento HTML (blue, azul).



- Las reglas de estilo tienen dos componentes: el selector y la declaración. Si en la declaración hay varias propiedades con sus valores, deben separarse con punto y coma(;).

Ejemplo:

```
h1  
{  
  color:orange;  
  text-align:center;  
}
```

Validación CSS:



3. DÓNDE APLICAR LAS REGLAS

- **CSS Directas:** Directamente a la **marca**.
- **CSS Indirectas dentro del documento:** En el **head** de la página.
- **CSS Indirectas fuera del documento:** Agrupar las reglas de estilo en un **archivo independiente** con extensión ***.css**

3. DÓNDE APLICAR LAS REGLAS

CSS Directas

Las CSS directas se usan para poder aplicar CSS a una parte del texto concreta. Para ello debemos utilizar la etiqueta “style”.

```
<X style="atributo1:valor1; atributo2:valor2;">Texto al que  
se le aplicará el estilo</X>
```

- “X” sería la etiqueta html a la que se le ha aplicado el estilo, mediante el parámetro “style”.
- El estilo está definido por unos atributos con un valor (“atributo1:valor1”), separados entre ellos por punto y coma.

3. DÓNDE APLICAR LAS REGLAS

```
<h1 style="color:blue; text-align:center;">
```

- Es la forma más fácil pero menos recomendada para aplicación de un estilo a una marca HTML.
- Cuando definimos estilos directamente en las marcas HTML, tenemos que tener en cuenta que el estilo se aplica únicamente a la marca donde indicamos la propiedad **style**, es decir, si tenemos dos secciones **h1**, deberemos definir la propiedad **style** para cada marca.

3. DÓNDE APLICAR LAS REGLAS

CSS Indirectas dentro del documento

- Se aplican cuando queremos que los estilos de las CSS se apliquen de una forma global a todas las etiquetas de un documento.
- Por ejemplo, queremos que todo el cuerpo del documento, el “**body**”, tenga un determinado tipo de letra, que los párrafos tengan un color determinado, que las cabeceras tengan un tamaño igual entre ellas, etc. Para conseguir estos propósitos, las CSS indirectas son ideales.
- Para incluir CSS indirecta dentro del mismo documento, utilizaremos la etiqueta **<style>**. La colocaremos en la cabecera del documento. Entre esta etiqueta y su cierre (**</style>**) escribiremos todos los estilos que queramos definir en el documento.

3. DÓNDE APLICAR LAS REGLAS

```
<head>
  <style type="text/css">
    hr {
      color: blue;
    }
    h3 {
      color: red;
      text-align: left;
    }
  </style>
</head>
```

- A la etiqueta “style” le hemos puesto el atributo **type=“text/css”**. Este atributo no es necesario, pero sí muy recomendable, pues le está indicando al navegador el tipo de sintaxis que usa la hoja de estilo.

3. DÓNDE APLICAR LAS REGLAS

CSS Indirectas fuera del documento

- Esta forma de aplicar CSS es útil si queremos aplicar los mismos estilos a diferentes documentos html.
- Deberemos crear un documento de extensión .css donde estarán determinados los diferentes estilos. Y desde cada página en la que queramos aplicarlos, habrá que indicar que debe buscar los estilos en ese documento.
- Para ello, debemos incluir la siguiente etiqueta dentro del **<head>**:

```
<link rel="stylesheet" type="text/css" href="urlhojadeestilos.css">
```

3. DÓNDE APLICAR LAS REGLAS

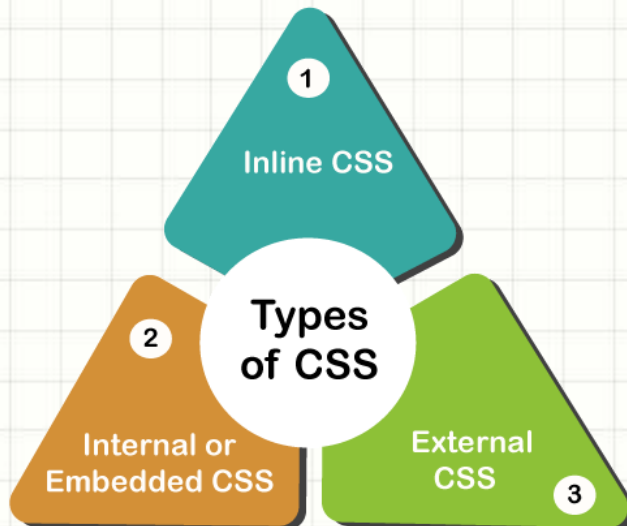
```
<head>  
  <link rel="stylesheet" type="text/css" href="mystyle.css">  
</head>
```

- El atributo “**rel**” debe ir con el valor “**stylesheet**” ya que está definiendo que “link” se está usando para insertar una hoja de estilo.
- El atributo “**type**” con valor “**text/css**” indica que el tipo de sintaxis que usa la hoja de estilo es CSS.
- El atributo “**href**” especificará la ubicación y el nombre de la hoja de estilo.

ORDEN APLICACIÓN DE ESTILOS

1. Opciones por defecto del navegador.
2. CSS indirectas fuera del documento.
3. CSS indirectas dentro del documento.
4. CSS directas a nivel de marca.

Los estilos se irán aplicando ***unos sobre otros***.



[Listado completo de propiedades CSS](#)

Ejemplo práctico 1

Ya sabemos el orden el que se aplican los estilos según la fuente dónde estén definidas las reglas CSS.

Crea una página web con toda su estructura básica y con ella un elemento cualquiera.

Ve añadiendo en el presente orden reglas de estilo personalizadas y observa como se van superponiendo:

1. Desde un fichero externo.
2. CSS indirecta dentro del documento.
3. CSS directa dentro de la etiqueta del elemento.

4. COLORES

- **Hexadecimal (#RRGGBB):** Se define con una almohadilla seguida de las cantidades de color para rojo, verde y azul. Las cantidades se expresan en hexadecimal (es decir, dos dígitos de 0-9 y A-F) .
De esta manera negro es #000000, rojo es #FF0000, gris oscuro es #333333 y blanco #FFFFFF.
Existen programas que te calculan estas cantidades a partir de un color que tú le das.
- **Hexadecimal abreviado (#RGB):** Si en el caso anterior los dos dígitos para rojo, verde y azul son los mismos (por ejemplo DD, BB o 22) se puede abreviar dejando sólo uno.
De esta manera negro es #000, rojo es #F00, gris oscuro es #333 y blanco #FFF.

4. COLORES

- **Combinaciones predefinidas en inglés:** Existen una serie de colores simples (hasta 140) que ya vienen predefinidos y que podemos usar con sus nombres en inglés.

Ejem: Red, Blue, Salmon

[Lista completa](#)

- **Cantidades de color en RGB:** `rgb(ROJO, VERDE, AZUL)`. Con esta función podemos indicar el color directamente con sus cantidades de rojo, verde y azul con números decimales de 8 bits (del 0 al 255).

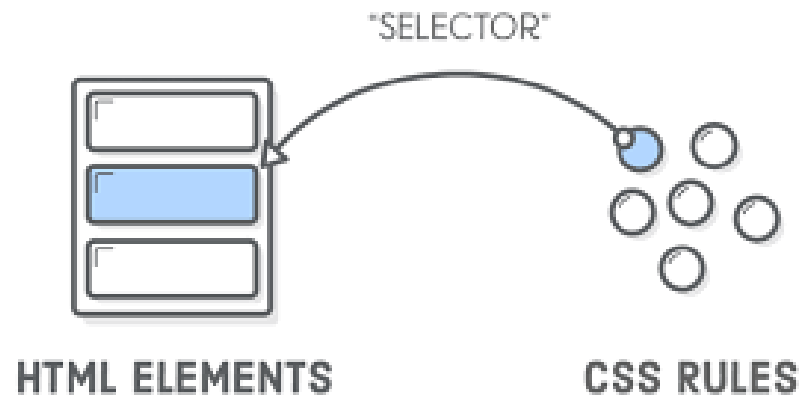
De esta manera el negro es `rgb(0,0,0)`; rojo es `rgb(255,0,0)`.

[Guía sobre colores](#)

[Diseño esquema de colores](#)

5. SELECTORES

- ▶ En CSS los **selectores** se utilizan para delimitar los elementos HTML de nuestra página web a los que queremos aplicar estilo. Hay una amplia variedad de selectores CSS, lo que permite una gran precisión a la hora de seleccionar elementos a los que aplicar estilo.



5.1 SELECTOR DE ETIQUETA

- Se utiliza para saber a qué elemento se le aplicará el formato:

```
<style>
  p {
    color: red;
  }
  h1, h2, h3 {
    color: #8A8E27;
    font-weight: normal;
    font-family: Arial, Helvetica;
  }
</style>
```

- En este caso cambiará el estilo de todos los p, h1, h2 y h3.

5.2 SELECTOR DE ID

- Debemos asignar una identificación única al elemento.

Por ejemplo:

```
<p id="destacado">Esto es un párrafo de prueba</p>
```

Para asignarle estilo:

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

- Observa el atributo `id` y `#`. El atributo `id` deberá ser único para cada etiqueta. De esta forma solo podremos aplicar una regla a un elemento.

5.3 SELECTOR DE CLASE

- Su finalidad es agrupar los elementos por clases para así distinguirlos de los demás.

Por ejemplo:

```
  
  

```

Para asignarle estilo:

```
.fotos {  
    border-width: 1px  
}
```

- Observa el atributo **class** y el **.** (punto)
- Al contrario que con id, con una sola regla podremos cambiar el estilo de varios elementos siempre que compartan la clase.

5.4 PSEUDOCCLASES

- Agrupa los elementos en función de los eventos que les haya ocurrido.

Pseudo-clase	Significado
:link	Vínculos no visitados
:visited	Vínculos visitados
:hover	Elementos con el cursor del ratón
:active	Elementos a los que se les está haciendo click
:focus	Elementos con el foco (se puede cambiar con el tabulador)

- Por ejemplo, si se quiere que los vínculos visitados aparezcan en azul:

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

5.4 PSEUDOCCLASES

- El siguiente ejemplo muestra cómo ocultar el subrayado cuando el usuario pasa el ratón por encima de cualquier enlace de la página:

```
a:hover { text-decoration: none; }
```



5.5 PSEUDOELEMENTOS

- ▶ Son partes de elementos de una página sin identidad propia y no pueden ser seleccionados por HTML, pero sí identificados y formateados por CSS.
- ▶ Por ejemplo, no existe etiqueta para marcar la primera letra de cada párrafo. Ejemplo:

```
/* :first-line permite aplicar estilos a la primera línea  
de un texto. */  
p:first-line {  
    text-transform: uppercase;  
}
```

```
/* :first-letter permite aplicar estilos a la primera  
letra del texto. */  
p:first-letter {  
    color: red;  
    font-size: 18px;  
}
```

Nota: para **comentarios** en CSS ->

[Listado de pseudoelementos](#)

```
/* esto es un comentario */
```

5.6 SELECTOR UNIVERSAL

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

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

- ▶ A pesar de su sencillez, no se utiliza habitualmente, ya que es difícil que un mismo estilo se pueda aplicar a todos los elementos de una página.

5.8 AGRUPACIÓN DE SELECTORES

- ▶ Agrupar selectores: Se aplican las mismas reglas de estilo a más de un elemento.

```
td, p {color: beige;}
```

- ▶ Combinar selectores: Se utiliza para obtener una selección de elementos más fina:

```
div.avisos span.especial { ... }
```

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

6. PROPIEDADES Y VALORES

- ▶ Para saber utilizar CSS hay que saber cómo seleccionar el elemento y aplicarle el formato, o sea, manejar el par **{propiedad:valor}**.
- ▶ Cada propiedad CSS posee reglas diferentes sobre los valores que puede aceptar:
 - De una lista predefinida.
 - Números, porcentajes, colores,...
 - Más de un tipo de valor.
- ▶ Hay propiedades que exigen que se especifique el valor de una forma determinada:
 - No dejar espacios entre número y unidad.
 - Utilizar unidades de medida específicas.
 - Especificar url de otro archivo.

6.1 IMPORTANCIA PRECEDENCIA

En ocasiones puede ocurrir que se definan **dos reglas** para un mismo elemento. A continuación se muestran los casos con los que se resuelve:

- ▶ **Herencia:** los descendientes de los elementos definidos por el selector heredan algunas propiedades del selector, y pueden aparecer efectos no deseados. Por ejemplo, la propiedad color se hereda mientras que la propiedad **border** no. Por lo cual hay que forzar la herencia con el valor **inherit**.

```
{border:inherit;}
```

Propiedades heredadas

Ejemplo, definimos la siguiente regla de estilo:

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

y el código HTML siguiente:

```
<p>Este párrafo tiene <em>texto enfatizado</em> en su interior.</p>
```

las palabras "texto enfatizado" aparecerán en verde, ya que el elemento **em** ha heredado el valor de la propiedad color a partir del elemento **p**. No recoge el valor inicial de la propiedad (que es el color usado por el elemento raíz cuando la página especifica que no hay color).

Propiedades no heredadas

Ejemplo, definimos la siguiente regla de estilo:

```
p { border: medium solid }
```

y el código HTML siguiente:

```
<p>Este párrafo tiene <em>texto enfatizado</em> en su interior.</p>
```

las palabras "texto enfatizado" no tendrán borde (ya que el valor inicial del estilo de borde es none).

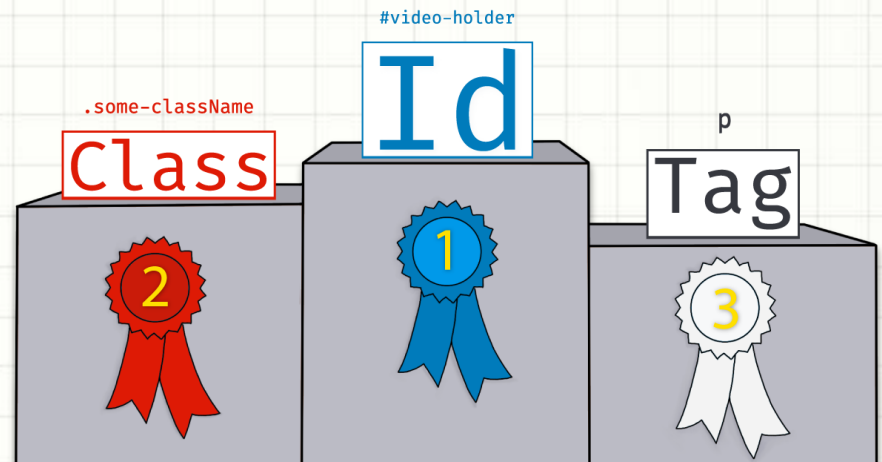
6.1 IMPORTANCIA PRECEDENCIA

- ▶ **Especificidad:** cuando al mismo elemento se le aplican dos reglas diferentes prevalece la que tenga el selector más concreto. Por ejemplo, un selector id prevalece sobre uno de tipo class, y éste sobre un selector simple.
- ▶ **Ubicación:** Las reglas que aparecen en último lugar tienen más peso. Por ejemplo, las reglas aplicadas localmente tienen más peso que las definidas de forma interna.

Ejemplo práctico

Ya sabemos el orden el que se aplican los estilos. Crea una página web con toda su estructura básica y un elemento que tenga una id y una clase. Ve añadiendo en el presente orden reglas de estilo personalizadas y observa su comportamiento:

1. Etiqueta
2. Clase
3. Id



7. UNIDADES DE MEDIDA

- ▶ Las medidas en CSS se emplean, entre otras, para definir la altura, anchura y márgenes de los elementos y para establecer el tamaño de letra del texto. Todas las medidas se indican como un valor numérico entero o decimal seguido de una unidad de medida (sin ningún espacio en blanco entre el número y la unidad de medida).
- ▶ CSS divide todas las unidades de medida en dos grupos: **absolutas** y **relativas**. Las medidas relativas definen su valor en relación con otra medida, por lo que para obtener su valor real, se debe realizar alguna operación con el valor indicado. Las unidades absolutas establecen de forma completa el valor de una medida, por lo que su valor real es directamente el valor indicado.

7.1 UNIDADES ABSOLUTAS

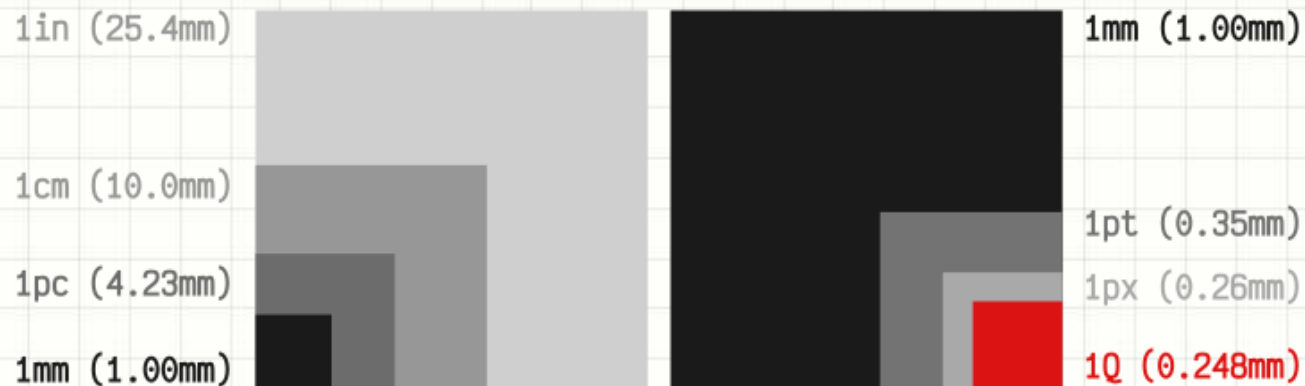
- ▶ Las unidades **absolutas** definen las medidas de forma completa, ya que sus valores reales no se calculan a partir de otro valor de referencia, sino que son directamente los valores indicados.
 - in, del inglés "inches", pulgadas (1 pulgada = 2.54 centímetros)
 - cm, centímetros
 - mm, milímetros
 - pt, puntos (1 punto equivale a 1 pulgada/72, es decir, unos 0.35mm)
 - pc, picas (1 pica equivale a 12 puntos, es decir, unos 4.23mm)

7.1 UNIDADES ABSOLUTAS

► Por ejemplo:

```
body { margin: 0.5in; }  
h1 { line-height: 2cm; }  
p { word-spacing: 4mm; }  
a { font-size: 12pt }  
span { font-size: 1pc }
```

Unidades absolutas



7.2 UNIDADES RELATIVAS

Las unidades **relativas** son más flexibles que las unidades absolutas porque se adaptan más fácilmente a los diferentes medios:

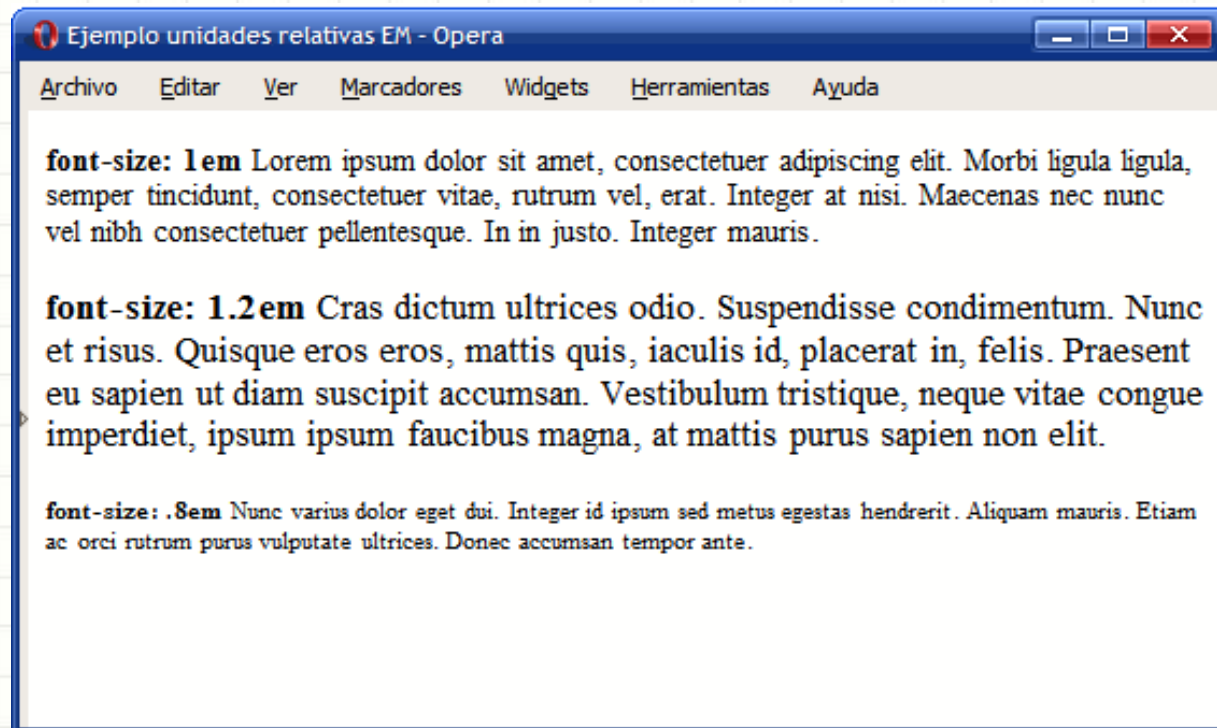
Unidad	Símbolo	Ejemplo	Descripción
Porcentaje	%	<code>#menu1 {width: 50%;}</code>	Porcentaje relativo al elemento contenedor.
Relativa al tamaño de letra	em	<code>#menu1 {font-size: 2.65em;}</code>	Tantas veces el tamaño que sea de aplicación como se indique. Por ejemplo si el tamaño de letra de aplicación es 12 píxeles, 1 em son 12px, 2 em son 24 px, 3 em son 36 px, etc.
Relativo a la x minúscula	ex	<code>#menu1 {font-size: 2.65ex;}</code>	Tantas veces la altura de la letra x minúscula como se especifique. Por ejemplo si la x minúscula que se debería mostrar mide 10mm, 1 ex son 10 mm, 1.5ex son 15mm, 2ex son 20mm, etc.
Pixel*	px	<code>#menu1 {font-size: 24px;}</code>	Tantas veces los puntos visibles que tenga la pantalla donde se visualice la página web como se especifique. *Puede considerarse una unidad híbrida, ni absoluta ni relativa.

7.2 UNIDADES RELATIVAS

- ▶ Por ejemplo, tamaño de letra del texto de la página al 90% del tamaño por defecto (depende del navegador):

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

- ▶ Ejemplo de **em** en diferentes párrafos:



7.3 MEZCLA DE AMBAS

Las distintas unidades se pueden mezclar entre los diferentes elementos de una misma página, por ejemplo:

```
body { font-size: 10px; }
```

```
h1 { font-size: 2.5em; }
```

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

Unidades relativas

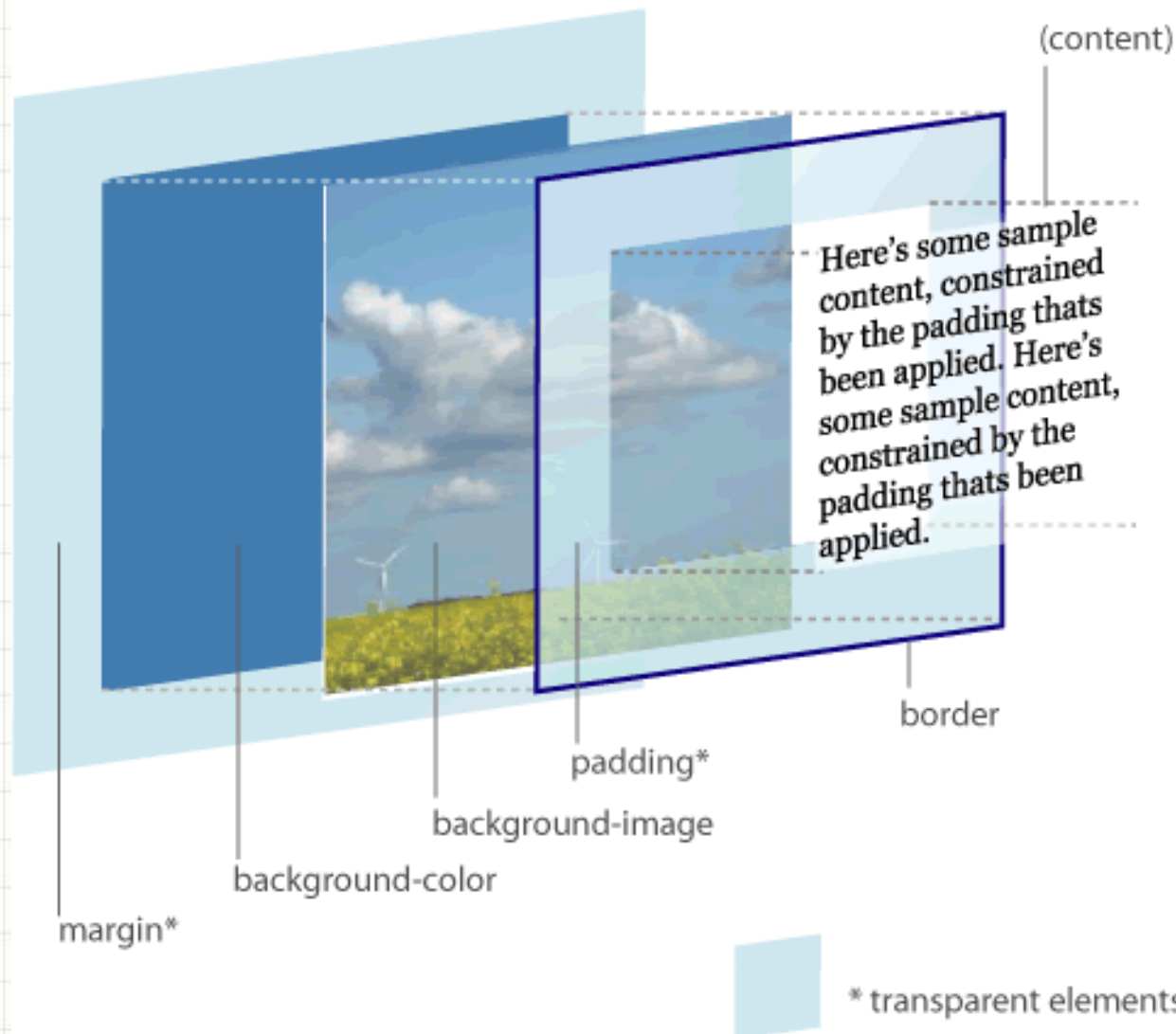


8. LAS CAPAS

- ▶ El aspecto más importante cuando se aplican estilos con CSS a una página web es su estructura; si en ella se etiquetan adecuadamente sus contenidos mediante clases e identificadores, se creará una estructura correcta y se facilitará el trabajo posterior.
- ▶ Para dotar a una página web con una buena estructura se utilizan las capas (cajas), las cuales son divisiones dentro del documento a través de la inclusión de elementos **div**. Esta etiqueta divide el documento en grandes secciones con un comportamiento independiente entre ellas.

8. LAS CAPAS

THE CSS BOX MODEL HIERARCHY



8.1 PARTES DE LAS CAJAS

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

- **Contenido (content):** se trata del contenido HTML del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)
- **Relleno (padding):** espacio libre opcional entre el contenido y el borde que lo encierra.
- **Borde (border):** línea que encierra completamente el contenido y su relleno.
- **Imagen de fondo (background image):** imagen que se muestra por detrás del contenido y el espacio de relleno.
- **Color de fondo (background color):** color que se muestra por detrás del contenido y el espacio de relleno.
- **Margen (margin):** espacio libre entre la caja y las posibles cajas adyacentes.

8.1 PARTES DE LAS CAJAS



8.2 Anchura y Altura

Con estas propiedades establecemos la anchura y altura de un elemento.

Propiedad	Valor	Descripción
width	<medida> <porcentaje> auto inherit	Establece la anchura de un elemento
height	<medida> <porcentaje> auto inherit	Establece la altura de un elemento

```
#lateral { width: 200px; }  
<div id="lateral">...</div>
```

```
#cabecera { height: 200px; }  
<div id="cabecera">...</div>
```

8.3 Margen

Los márgenes se utilizan para crear espacio alrededor de los elementos, fuera de los bordes definidos.

Propiedad	Ejem.Valor	Descripción
margin-top	20px	Estable el margen superior de un elemento
margin-right	1cm	Estable el margen derecho de un elemento
margin-bottom	10%	Estable el margen inferior de un elemento. En el ejemplo el margen inferior es un 10% del elemento contenedor.
margin-left	inherit	Estable el margen izquierdo de un elemento. En el ejemplo el margen izquierdo es heredado del elemento padre.
margin	25px 50px 75px 100px;	Top margin es 25px Right margin es 50px Bottom margin es 75px Left margin es 100px

8.4 Padding

El padding se utiliza para crear espacio alrededor del contenido de un elemento, dentro de los bordes definidos.

Propiedad	Ejem.Valor	Descripción
padding-top	20px	Estable el espacio por encima de un elemento
padding-right	1cm	Estable el espacio por la derecha de un elemento
padding-bottom	10%	Estable el espacio por debajo de un elemento. En el ejemplo el espacio inferior es un 10% del elemento contenedor.
padding-left	inherit	Estable el espacio por la izquierda de un elemento. En el ejemplo el espacio izquierdo es heredado del elemento padre.
padding	25px 50px 75px 100px;	Top padding es 25px Right padding es 50px Bottom padding es 75px Left padding es 100px

8.4 Padding

```
div {  
  border: 1px solid black;  
  background-color: lightblue;  
  padding-top: 40px;  
  padding-right: 20px;  
  padding-bottom: 40px;  
  padding-left: 20px;  
}
```

This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px, and a left padding of 80px.

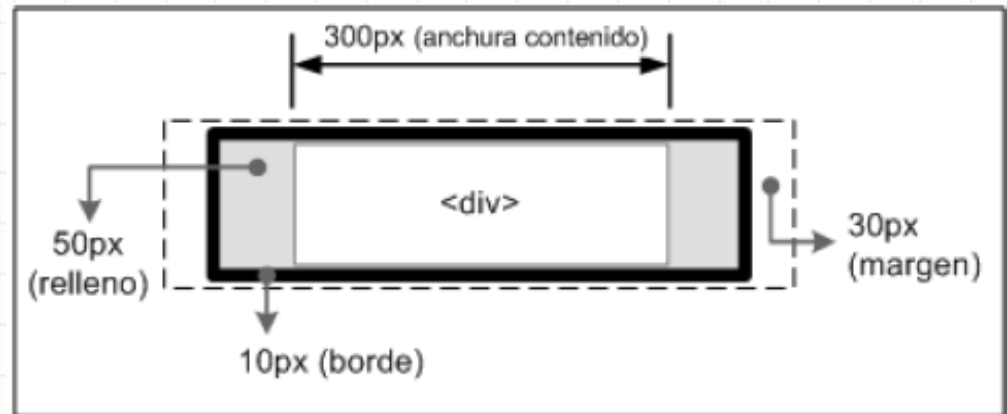
```
div {  
  margin: 40px;  
  border: 1px solid #4CAF50;  
}
```

This element has a margin of 40px.

8.4 Padding

La anchura y altura de un elemento no solamente se calculan teniendo en cuenta sus propiedades **width** y **height**. El margen, el relleno y los bordes establecidos a un elemento determinan la anchura y altura final del elemento.

```
div {  
  width: 300px;  
  padding-left: 50px;  
  padding-right: 50px;  
  margin-left: 30px;  
  margin-right: 30px;  
  border: 10px solid black;  
}
```



En el ejemplo, la anchura del elemento en pantalla sería igual a la suma de la anchura original, los márgenes, los bordes y los rellenos:

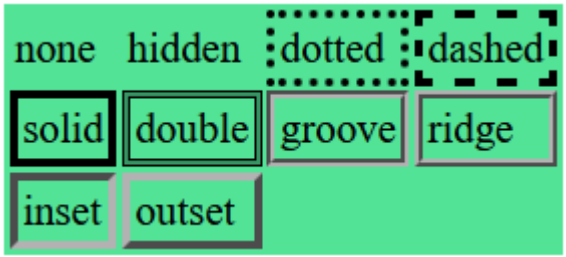
$$30\text{px} + 10\text{px} + 50\text{px} + 300\text{px} + 50\text{px} + 10\text{px} + 30\text{px} = 480 \text{ píxel}$$

8.5 Bordos

Los siguientes estilos se aplican a los bordes de un elemento.

Propiedad	Ejem.Valor	Descripción
border-color	#FF0000 Red	Permite indicar el color del borde del elemento al que se lo aplicamos. Este color se le indica con el modo RGB o con el nombre del color.
border-top-color border-right-color border-bottom-color border-left-color	#FF0000 Red	Permite indicar el color de cada borde individualmente: <ul style="list-style-type: none">• top: Borde superior• right: Borde derecho• bottom: Borde inferior• left: Borde izquierdo

8.5 Bordes

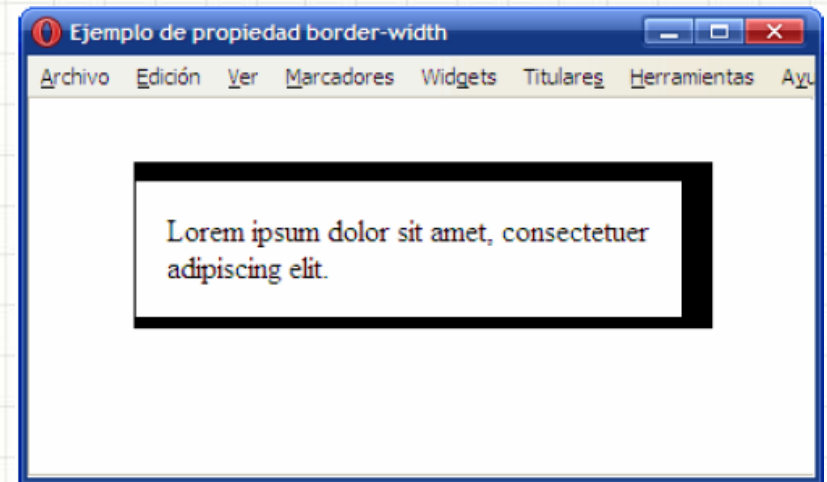
Propiedad	Ejem.Valor	Descripción
border-style 	none; hidden; dotted; dashed; solid; double; groove; ridge; inset; outset;	Permite indicar el estilo del borde del elemento al que se lo aplicamos. Se puede especificar usando uno, dos, tres o cuatro valores: <ul style="list-style-type: none"> • un valor: se aplica el mismo estilo a los cuatro lados • dos valores: el primer estilo se aplica a la parte superior e inferior, el segundo a la izquierda y a la derecha • tres valores, el primer estilo se aplica a la parte superior, el segundo a la izquierda y derecha, el tercero a la parte inferior. • cuatro valores, los estilos se aplican a la parte superior, derecha, inferior e izquierda en ese orden.
border-top-style border-right-style border-bottom-style border-left-style		Permite indicar el estilo de cada borde individualmente: <ul style="list-style-type: none"> • top: Borde superior • right: Borde derecho • bottom: Borde inferior • left: Borde izquierdo

8.5 Bordos

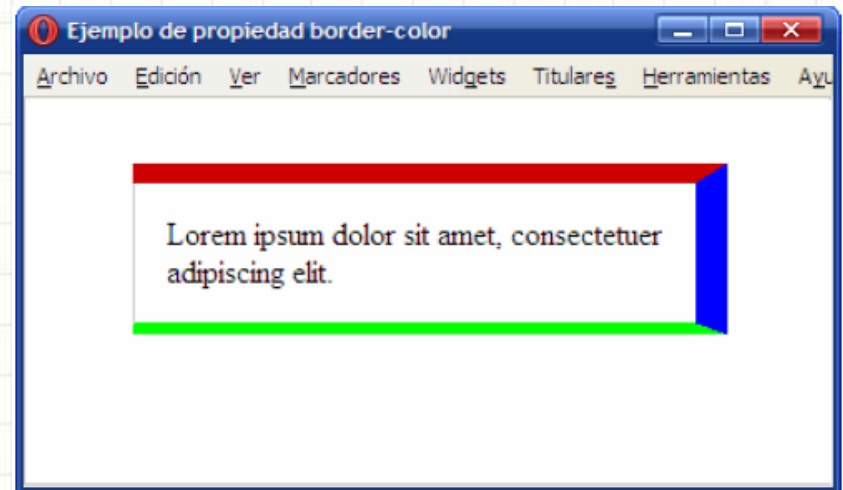
Propiedad	Ejem.Valor	Descripción
border-width	5px thin medium thick	Nos permite indicar el tamaño del borde del elemento al que se lo aplicamos. El tamaño se lo debemos indicar con alguna de las unidades CSS. El ancho se puede establecer como un tamaño específico (en px, pt, cm, em, etc.) o usando uno de los tres valores predefinidos: thin , medium , thick .
border	1px solid #000	Utilizado sólo, sirve para establecer los atributos que le indiquemos a los cuatro bordes del elemento al que se lo aplicamos. Podemos especificarle un “width”, un “style “ y un “color” y éstos se aplicarán a los cuatro bordes del elemento.
border-radius	12px	Nos permite añadir bordes redondeados al elemento.
border-collapse	collapse separate	Determina como se mostrarán y se comportarán los bordes de elementos contiguos en las tablas. Este estilo admite dos valores: <ul style="list-style-type: none">• collapse: indica a los bordes CSS que deben solaparse, mostrando uno sólo.• separate: muestra los dos bordes contiguos, sin que éstos se solapen.

8.5 Bordres

```
div {  
  border-top-width: 10px;  
  border-right-width: 1em;  
  border-bottom-width: thick;  
  border-left-width: thin;  
}
```



```
div {  
  border-top-color: #CC0000;  
  border-right-color: blue;  
  border-bottom-color: #00FF00;  
  border-left-color: #CCC;  
}
```



8.6 Estilos de fuente

Los siguientes estilos se pueden aplicar a las fuentes. Podemos cambiar el tamaño del texto, el color del mismo, su anchura, su estilo e incluso su tipo de letra.

Propiedad	Ejem.Valor	Descripción
color	#FF0000 Red	Define el color del texto. Lo podemos indicar como nombre del color o con el modo RGB.
font-size	x-large medium 25px	Define el tamaño de la letra. Se lo podemos indicar utilizando las unidades de medida de CSS o mediante los siguientes valores: xx-large , x-large , large , medium , small , x-small , xx-small
font-family	Georgia	Define el tipo de fuente que queramos que tenga el texto. Si no le indicamos nada, el texto estará escrito en la tipografía que el usuario tenga en su sistema.
font-weight	bold bolder lighter 100	Define la anchura que tendrá cada carácter. La anchura la podemos especificar de forma relativa al valor actual que tenga el texto (bold , bolder , lighter) o mediante un valor numérico (del 100 al 900).
font-style	normal italic oblique	Define el estilo al texto. Los estilos que podemos aplicar al texto son: normal , italic y oblique .

8.7 Estilos de texto

Los siguientes estilos se aplican a los párrafos. Podemos cambiar el alto de la línea, darle una decoración al párrafo, alinearlos, etc..

Propiedad	Ejem.Valor	Descripción
text-decoration	underline overline line-through	Con este estilo podremos subrayarlo (underline), sobrerayarlo (overline) o tacharlo (line-through).
text-align	left right center justify	Define la alineación del texto: izquierda (left), derecha (right), entrado (center) o justificado (justify).
text-indent	15px	Con este estilo podemos hacer sangrados o márgenes en los párrafos. Para ello debemos indicarle un valor numérico que será el tamaño del sangrado.
text-transform	uppercase lowercase capitalize	Este estilo nos permite cambiar la apariencia del párrafo: cambiar todo el texto a mayúsculas (uppercase), todo el texto a minúsculas (lowercase) o hacer que todas las primeras letras del párrafo se conviertan en mayúsculas (capitalize).

8.8 Estilos de fondo

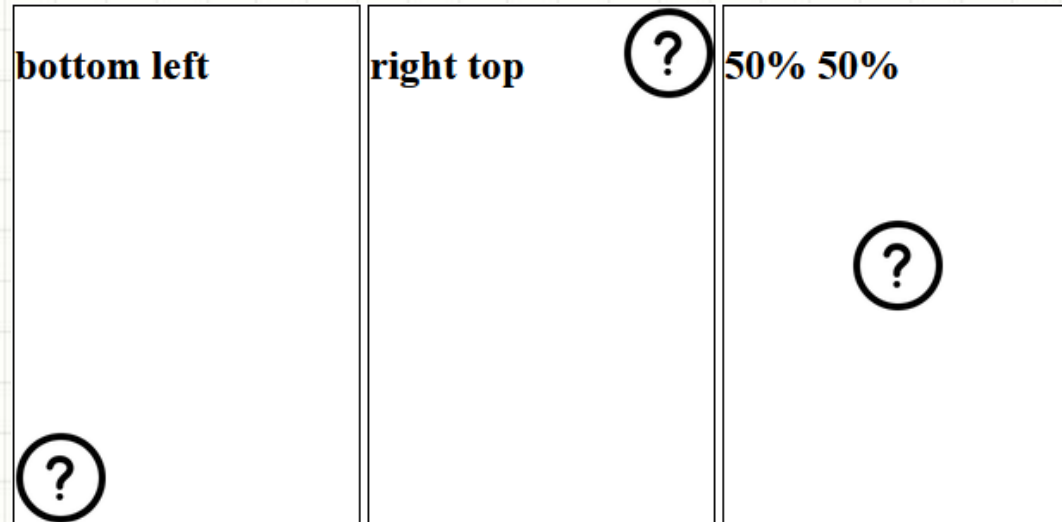
Los siguientes estilos se aplican al fondo de pantalla. Podemos cambiar el color, la imagen de fondo, la posición, etc..

Propiedad	Ejem.Valor	Descripción
background-color	#FF0000 Red transparent	Con este estilo podremos cambiar el color de fondo de la etiqueta.
background-image	url(fondo.jpg)	Con este estilo podremos aplicar una imagen de fondo a la etiqueta.
background-repeat	repeat repeat-x repeat-y no-repeat	Se usa para especificar si la imagen se repite: repeat: se repite en horizontal y vertical repeat-x: solo se repite en horizontal repeat-y: solo se repite en vertical no-repeat: no se repite
background-position	top, center, bottom left, center, right	Este atributo se usa para especificar la posición de la imagen de fondo: verticalmente (top , center y bottom) y horizontalmente (left , center y right). Si expresamos dos coordenadas, la primera será la horizontal y la segunda la vertical.

8.8 Estilos de fondo

```
#caja1 {  
  width: 300px;  
  height: 500px;  
  border: 1px solid black;  
  background-image: url("help.png");  
  background-repeat: no-repeat;  
  background-position: bottom left;  
  display: inline-block;  
}  
#caja2 {  
  width: 300px;  
  height: 500px;  
  border: 1px solid black;  
  background-image: url("help.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
  display: inline-block;  
}  
#caja3 {  
  width: 300px;  
  height: 500px;  
  border: 1px solid black;  
  background-image: url("help.png");  
  background-repeat: no-repeat;  
  background-position: 50% 50%;  
  display: inline-block;  
}
```

```
<div id="caja1"><h2>bottom left</h2></div>  
<div id="caja2"><h2>right top</h2></div>  
<div id="caja3"><h2>50% 50%</h2></div>
```



9. Propiedades abreviadas

CSS define una serie de propiedades que permiten establecer de forma directa los valores que se quieren aplicar. Se denominan propiedades de tipo “**shorthand**”. Todas las propiedades permiten indicar entre uno y cuatro valores:

- Si se indica **1 valor**, se aplica a los cuatro bordes.
- Si se indican **2 valores**, el primero se aplica al borde superior e inferior y el segundo valor se aplica al borde izquierdo y derecho.
- Si se indican **3 valores**, el primero se aplica al borde superior, el segundo al borde izquierdo y derecho y el tercero al borde inferior.
- Si se indican **4 valores**, el primero se aplica al borde superior, el segundo al borde derecho, el tercero al borde inferior y el cuarto al borde izquierdo.

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

10. POSICIONAMIENTO

Utilizando las **propiedades** que proporciona CSS para alterar la posición de las cajas es posible realizar efectos avanzados utilizando **position**:

- **static**: Se ignoran los valores de las propiedades top, right, bottom y left que se verán a continuación.
- **relative**: El desplazamiento de la caja se controla con las propiedades top, right, bottom y left.
- **absolute**: El desplazamiento de la caja también se controla con las propiedades top, right, bottom y left, pero su interpretación es mucho más compleja, ya que el origen de coordenadas del desplazamiento depende del valor de la propiedad position de su elemento contenedor.
- **fixed**: El desplazamiento se establece de la misma forma que en el posicionamiento absoluto, pero en este caso el elemento permanece inamovible en la pantalla.

position	Posicionamiento
Valores	<code>static</code> <code>relative</code> <code>absolute</code> <code>fixed</code> <code>inherit</code>
Se aplica a	Todos los elementos
Valor inicial	<code>static</code>
Descripción	Selecciona el posicionamiento con el que se mostrará el elemento

10. POSICIONAMIENTO

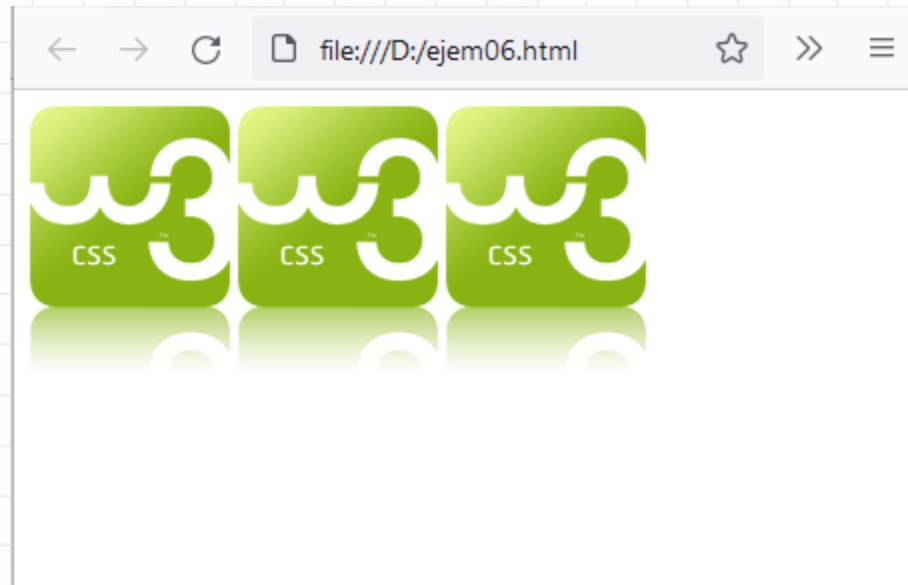
Normalmente, cuando se posiciona una caja también es necesario desplazarla respecto de su posición original o respecto de otro origen de coordenadas. CSS define cuatro propiedades llamadas ***top***, ***right***, ***bottom*** y ***left*** para controlar el desplazamiento de las cajas posicionadas.

top right bottom left	Desplazamiento superior Desplazamiento lateral derecho Desplazamiento inferior Desplazamiento lateral izquierdo
Valores	<medida> <porcentaje> auto inherit
Se aplica a	Todos los elementos posicionados
Valor inicial	auto
Descripción	Indican el desplazamiento horizontal y vertical del elemento respecto de su posición original

10.1 Posicionamiento normal

El posicionamiento normal o estático es el modelo que utilizan por defecto los navegadores para mostrar los elementos de las páginas.

En este modelo, ninguna caja se desplaza respecto de su posición original.



10.2 Posicionamiento relativo

El posicionamiento relativo permite desplazar una caja respecto de su posición original establecida mediante el posicionamiento normal.

El desplazamiento de la caja se controla con las propiedades ***top***, ***right***, ***bottom*** y ***left***.

Aplicado al ejemplo anterior.

```
img.pos {  
    position: relative;  
    top: 8em;  
}
```

```
  
  

```


10.2 Posicionamiento relativo

El resto de imágenes no varían su posición y por tanto no ocupan el hueco dejado por la primera imagen, ya que el posicionamiento relativo no influye en el resto de elementos de la página.

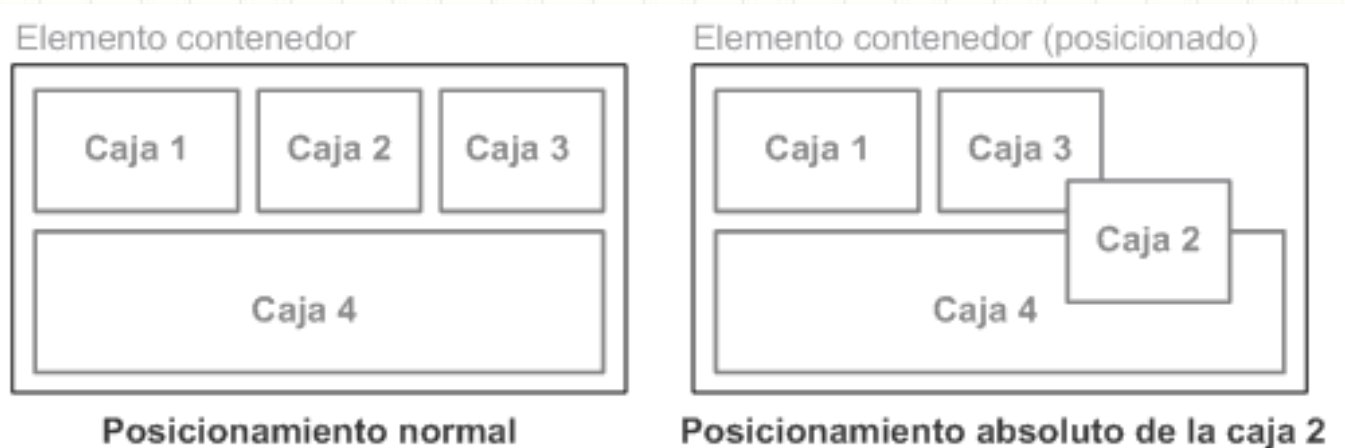
Hay que tener en cuenta que se pueden producir solapamientos con otros elementos de la página.



10.3 Posicionamiento absoluto

El posicionamiento absoluto se emplea para establecer de forma precisa la posición en la que se muestra la caja de un elemento. La nueva posición de la caja se indica mediante las propiedades ***top***, ***right***, ***bottom*** y ***left***.

Cuando una caja se posiciona de forma absoluta, el resto de elementos de la página la ignoran y ocupan el lugar original ocupado por la caja posicionada. Es probable que se produzcan solapamientos con otras cajas.



10.3 Posicionamiento absoluto

```
/* CSS */
div {
    border: 2px solid #CCC;
    padding: 1em;
    margin: 1em 0 1em 4em;
    width: 300px;
}
```

```
<!-- HTML -->
```

```
<div>
```

```
  
```

```
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Phasellus ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat
cursus, risus mi viverra augue, at pulvinar turpis leo sed orci.</p>
</div>
```

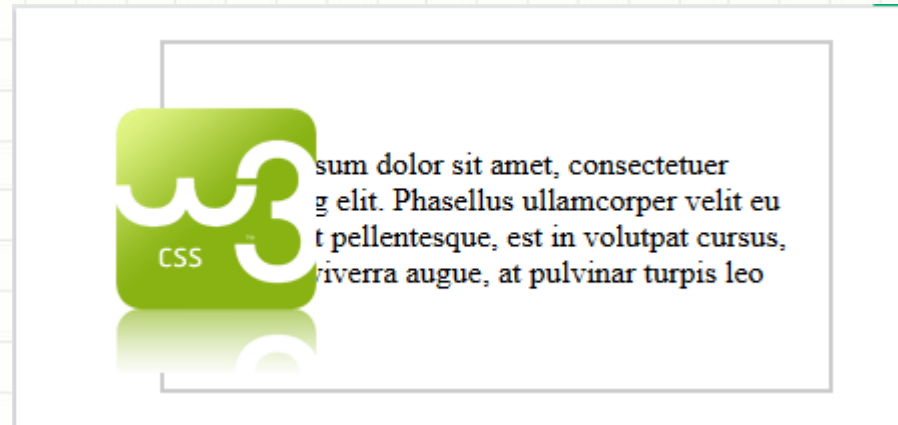


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus ullamcorper velit eu ipsum. Ut pellentesque, est in volutpat cursus, risus mi viverra augue, at pulvinar turpis leo sed orci.

10.3 Posicionamiento absoluto

En primer lugar, se posiciona de forma absoluta la imagen mediante la propiedad ***position*** y se indica su nueva posición mediante las propiedades ***top*** y ***left***:

```
div img {  
  position: absolute;  
  top: 50px;  
  left: 50px;  
}
```



La imagen posicionada de forma absoluta no toma como origen de coordenadas la esquina superior izquierda de su elemento contenedor **<div>**, sino que su referencia es la esquina superior izquierda de la página.

10.4 Posicionamiento fijo

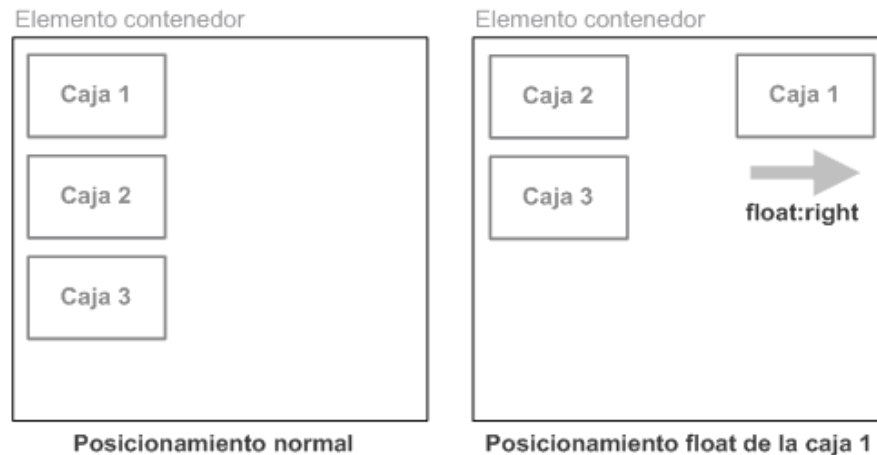
El estándar CSS considera que el posicionamiento fijo es un caso particular del posicionamiento absoluto, ya que sólo se diferencian en el comportamiento de las cajas posicionadas.

La principal característica de una caja posicionada de forma fija es que su posición es inamovible dentro de la ventana del navegador.

El posicionamiento fijo hace que las cajas no modifiquen su posición ni aunque el usuario suba o baje la página en la ventana de su navegador.

10.5 Posicionamiento flotante

El posicionamiento flotante es uno de los más utilizados. Cuando una caja se posiciona con el modelo de posicionamiento flotante, automáticamente se convierte en una caja flotante, lo que significa que se desplaza hasta la zona más a la izquierda o más a la derecha de la posición en la que originalmente se encontraba.

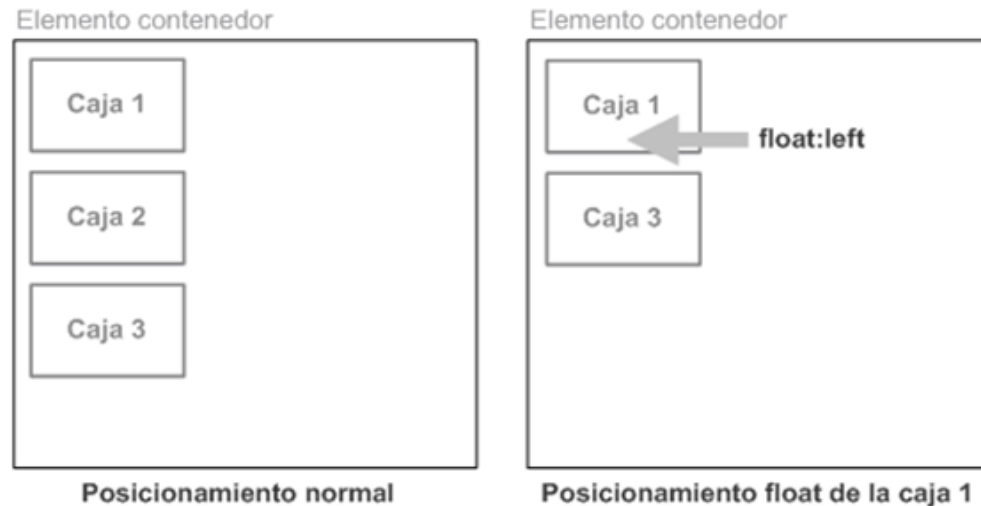


Cuando se posiciona una caja de forma flotante:

- La caja deja de pertenecer al flujo normal de la página, lo que significa que el resto de cajas ocupan el lugar dejado por la caja flotante.
- La caja flotante se posiciona lo más a la izquierda o lo más a la derecha posible de la posición en la que se encontraba originalmente.

10.5 Posicionamiento flotante

Si en el anterior ejemplo la caja 1 se posiciona de forma flotante hacia la izquierda:

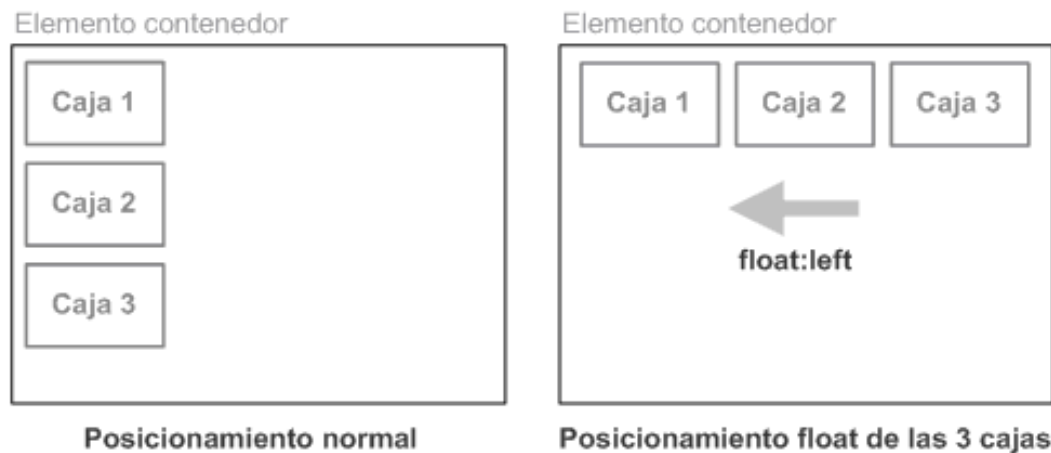


La caja 1 es de tipo flotante, por lo que desaparece del flujo normal de la página y el resto de cajas ocupan su lugar. El resultado es que la caja 2 ahora se muestra donde estaba la caja 1 y la caja 3 se muestra donde estaba la caja 2.

Al mismo tiempo, la caja 1 se desplaza todo lo posible hacia la izquierda de la posición en la que se encontraba. El resultado es que la caja 1 se muestra encima de la nueva posición de la caja 2 y tapa todos sus contenidos.

10.5 Posicionamiento flotante

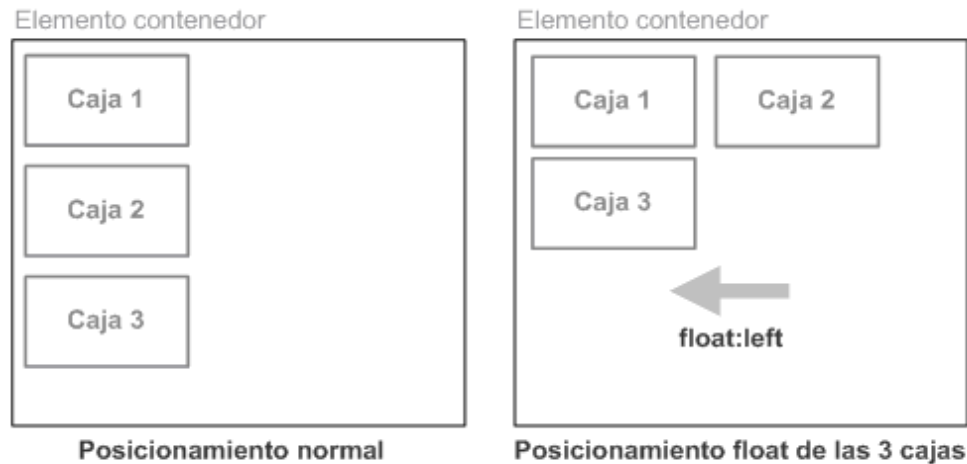
Si existen otras cajas flotantes, al posicionar de forma flotante una caja, se tiene en cuenta el sitio disponible. En el siguiente ejemplo se posicionan de forma flotante hacia la izquierda las tres cajas:



En el ejemplo anterior, las cajas no se superponen entre sí porque las cajas flotantes tienen en cuenta las otras cajas flotantes existentes. Como la caja 1 ya estaba posicionada lo más a la izquierda posible, la caja 2 sólo puede colocarse al lado del borde derecho de la caja 1, que es el sitio más a la izquierda posible respecto de la zona en la que se encontraba.

10.5 Posicionamiento flotante

Si no existe sitio en la línea actual, la caja flotante baja a la línea inferior hasta que encuentra el sitio necesario para mostrarse lo más a la izquierda o lo más a la derecha posible en esa nueva línea:



Las cajas flotantes influyen en la disposición de todas las demás cajas. Los elementos en línea hacen sitio a las cajas flotantes adaptando su anchura al espacio libre dejado por la caja desplazada. Los elementos de bloque no les hacen sitio, pero sí que adaptan sus contenidos para que no se solapen con las cajas flotantes.

10.5 Posicionamiento flotante

La propiedad CSS que permite posicionar de forma flotante una caja se denomina ***float***:

float	Posicionamiento float
Valores	left right none inherit
Se aplica a	Todos los elementos
Valor inicial	none
Descripción	Establece el tipo de posicionamiento flotante del elemento

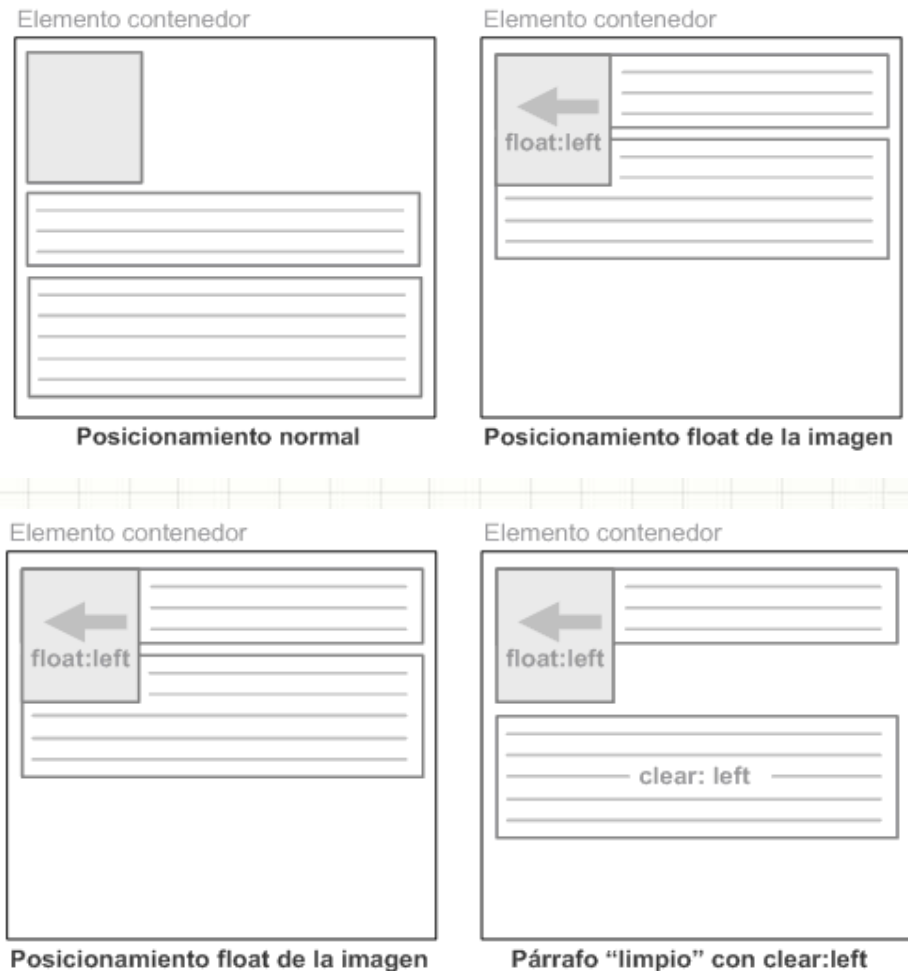
El valor ***none*** permite anular el posicionamiento flotante de forma que el elemento se muestre en su posición original.

Uno de los principales motivos para la creación del posicionamiento ***float*** fue precisamente la posibilidad de colocar imágenes alrededor de las cuales fluye el texto.

10.5 Posicionamiento flotante

CSS permite controlar la forma en la que los contenidos fluyen alrededor de los elementos posicionados mediante *float*.

De hecho, en muchas ocasiones es admisible que algunos contenidos fluyan alrededor de una imagen, pero el resto de contenidos deben mostrarse en su totalidad sin fluir alrededor de la imagen:



10.5 Posicionamiento flotante

```
<style>  
img {  
  float: right;  
}  
</style>
```

The float Property

In this example, the image will float to the right in the text, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac..



10.5 Posicionamiento flotante

La propiedad ***clear*** permite modificar el comportamiento por defecto del posicionamiento flotante para forzar a un elemento a mostrarse debajo de cualquier caja flotante:

clear	Despejar los elementos flotantes adyacentes
Valores	<code>none</code> <code>left</code> <code>right</code> <code>both</code> <code>inherit</code>
Se aplica a	Todos los elementos de bloque
Valor inicial	<code>none</code>
Descripción	Indica el lado del elemento que no debe ser adyacente a ninguna caja flotante

Se indica el lado del elemento HTML que no debe ser adyacente a ninguna caja posicionada de forma flotante.

- **Left:** el elemento no permite flotantes a su izquierda
- **Right:** el elemento no permite flotantes a su derecha
- **Both:** el elemento no permite flotantes a ningún lado

10.5 Posicionamiento flotante

```
<style>
img {
  float: right;
}
p.clear {
  clear: right;
}
</style>
....
<p class="clear">Praesent ...
```

The float Property

In this example, the image will float to the right in the text, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio.



Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac..

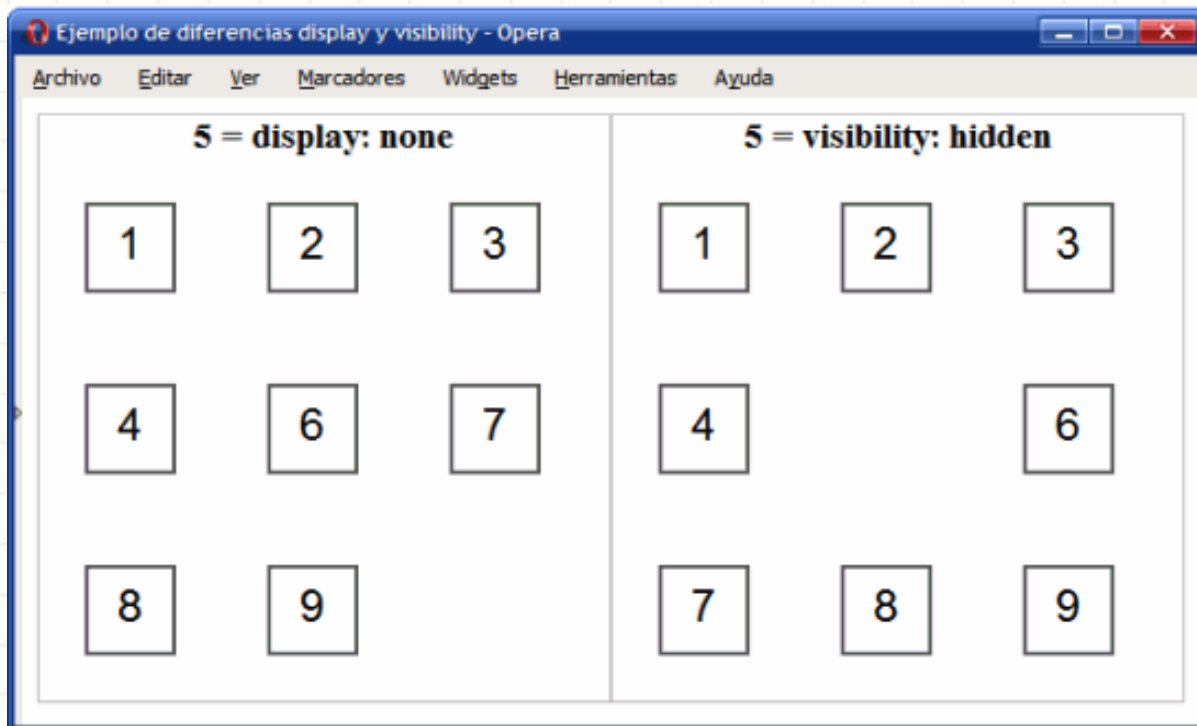
11. VISUALIZACIÓN: DISPLAY y VISIBILITY

Las propiedades ***display*** y ***visibility*** controlan la visualización de los elementos. Permiten ocultar elementos de la página. Habitualmente se utilizan junto con JavaScript para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes según convenga.

La propiedad ***display*** permite ocultar completamente un elemento 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.

11. VISUALIZACIÓN: DISPLAY y VISIBILITY

Por otra parte, la propiedad **visibility** permite hacer invisible un elemento, lo que significa que 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.



Diferencia entre la propiedad display y visibility (caja 5)

11. VISUALIZACIÓN: DISPLAY y VISIBILITY

En general, cuando se oculta un elemento no es deseable que siga ocupando sitio en la página, por lo que la propiedad **display** se utiliza mucho más que la propiedad **visibility**.

display	Visualización de un elemento
Valores	inline block none list-item run-in inline-block table inline-table table-row-group table-header-group table-footer-group table-row table-column-group table-column table-cell table-caption inherit
Se aplica a	Todos los elementos
Valor inicial	inline
Descripción	Permite controlar la forma de visualizar un elemento e incluso ocultarlo

- El valor **block** muestra un elemento como si fuera un elemento de bloque (con salto de línea), independientemente del tipo de elemento.
- El valor **inline** visualiza un elemento en forma de elemento en línea (sin salto de línea), independientemente del tipo.
- El valor **none** oculta un elemento y hace que desaparezca de la página.

[Ejemplo de comportamiento](#)

11. VISUALIZACIÓN: DISPLAY y VISIBILITY

Visibility sólo permite hacer visibles o invisibles a los elementos de la página. Inicialmente todas las cajas que componen la página son visibles.

Empleando el valor **hidden** es posible convertir una caja en invisible. El resto de elementos de la página se muestran como si la caja todavía fuera visible.

visibility	Visibilidad de un elemento
Valores	visible hidden collapse inherit
Se aplica a	Todos los elementos
Valor inicial	visible
Descripción	Permite hacer visibles e invisibles a los elementos

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.

Su efecto es similar al de la propiedad **display**, ya que oculta completamente la fila y/o columna y se pueden mostrar otros contenidos en ese lugar.

Si se utiliza el valor **collapse** sobre cualquier otro tipo de elemento, su efecto es idéntico al valor **hidden**.

11.1 Display, float y position

Cuando se establecen las propiedades ***display***, ***float*** y ***position*** sobre una misma caja, su interpretación es la siguiente.

1. Si ***display*** vale ***none***, se ignoran las propiedades float y position y la caja no se muestra en la página.
2. Si ***position*** vale ***absolute*** o ***fixed***, la caja se posiciona de forma absoluta, se considera que float vale none y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque. La posición de la caja se determina mediante el valor de las propiedades ***top***, ***right***, ***bottom*** y ***left***.
3. En cualquier otro caso, si ***float*** tiene un valor distinto de ***none***, la caja se posiciona de forma flotante y la propiedad display vale block tanto para los elementos en línea como para los elementos de bloque.

11.2 Overflow

En algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda. La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad ***width*** y/o ***height***.

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

overflow	Parte sobrante de un elemento
Valores	visible hidden scroll auto inherit
Se aplica a	Elementos de bloque y celdas de tablas
Valor inicial	visible
Descripción	Permite controlar los contenidos sobrantes de un elemento

11.2 Overflow

- **visible:** el contenido no se corta y se muestra sobresaliendo de la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- **hidden:** el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- **scroll:** solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de scroll que permiten visualizar el resto del contenido.
- **auto:** el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad scroll.

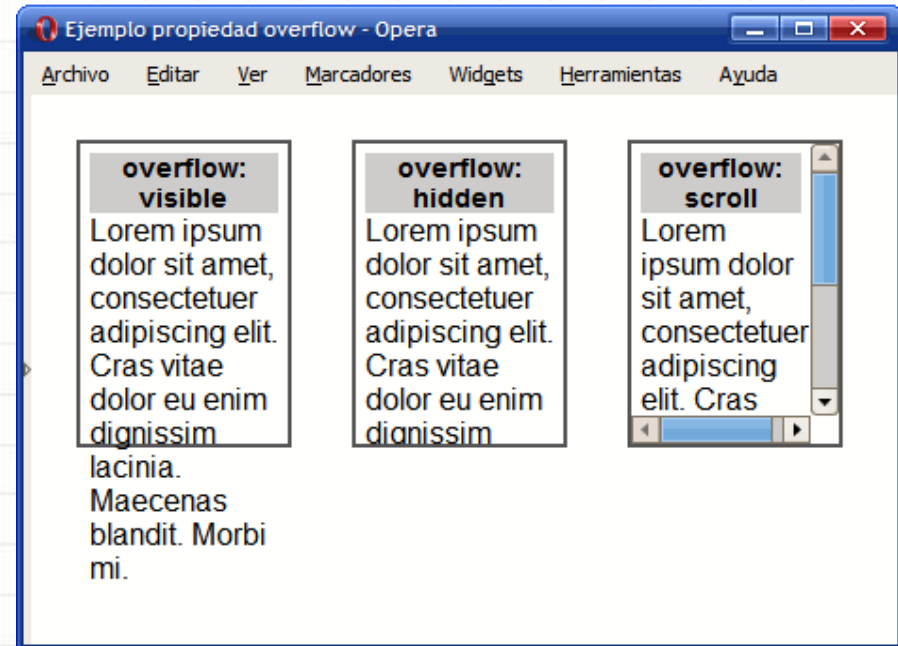
11.2 Overflow

```
div {  
  display: inline;  
  float: left;  
  margin: 1em;  
  padding: .3em;  
  border: 2px solid #555;  
  width: 100px;  
  height: 150px;  
  font: 1em Arial, Helvetica, sans-serif;  
}
```

```
<div><h3>overflow: visible</h3>Lorem ipsum...</div>
```

```
<div style="overflow:hidden"><h3>overflow: hidden</h3>Lorem  
ipsum...</div>
```

```
<div style="overflow:scroll"><h3>overflow: scroll</h3>Lorem  
ipsum...</div>
```



11.3 Z-Index

CSS permite controlar la posición tridimensional de las cajas posicionadas mediante la propiedad ***z-index***.

De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos. Así es posible crear páginas complejas con varios niveles o capas.

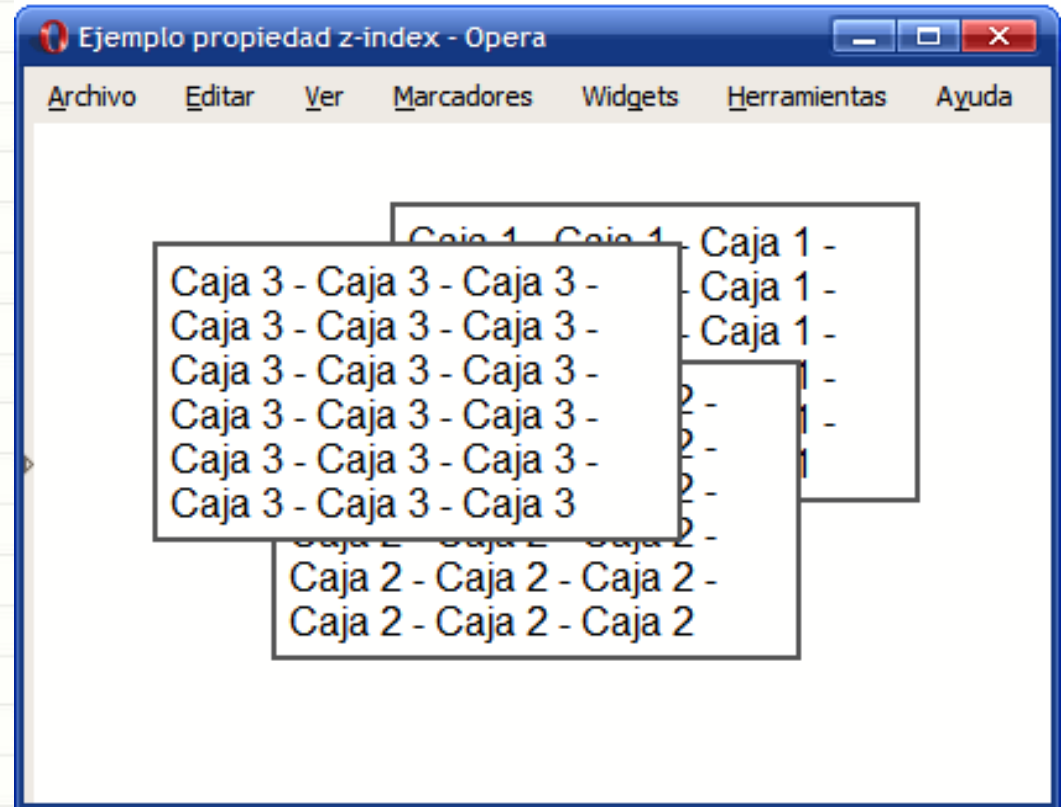
El valor más común de la propiedad ***z-index*** es un número entero.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja. Un elemento con z-index: 10 se muestra por encima de los elementos con z-index: 8 o z-index: 9

z-index	Orden tridimensional
Valores	auto <numero> inherit
Se aplica a	Elementos que han sido posicionados explícitamente
Valor inicial	auto
Descripción	Establece el nivel tridimensional en el que se muestra el elemento

11.3 Z-Index

```
div {  
  position: absolute;  
}  
#caja1 {  
  z-index: 5;  
  top: 1em;  
  left: 8em;  
}  
#caja2 {  
  z-index: 15;  
  top: 5em;  
  left: 5em;  
}  
#caja3 {  
  z-index: 25;  
  top: 2em;  
  left: 2em;  
}
```



11.3 Z-Index

La propiedad ***z-index*** sólo tiene efecto en los elementos posicionados, por lo que es obligatorio que la propiedad ***z-index*** vaya acompañada de la propiedad ***position***.

Si debes posicionar un elemento pero no quieres moverlo de su posición original ni afectar al resto de elementos de la página, puedes utilizar el posicionamiento relativo (***position: relative***).

12. CENTRAR HORIZONTALMENTE

A medida que aumenta el tamaño y la resolución de las pantallas, se hace más difícil diseñar páginas que se adapten al tamaño de la ventana del navegador. El principal reto que se presenta con resoluciones superiores a 1024x768 píxel, es que las líneas de texto son demasiado largas como para leerlas con comodidad. Por ese motivo, normalmente se opta por diseños con una anchura fija limitada a un valor aceptable para mantener la estética y la legibilidad del texto.

Por otra parte, los navegadores alinean por defecto las páginas web a la izquierda de la ventana. Cuando la resolución de la pantalla es muy grande, la mayoría de páginas de anchura fija alineadas a la izquierda parecen muy estrechas y provocan una sensación de vacío.

12. CENTRAR HORIZONTALMENTE

La solución más sencilla para evitar los grandes espacios en blanco consiste en crear páginas con una anchura fija adecuada y centrar la página horizontalmente respecto de la ventana del navegador. La siguiente imagen muestran el aspecto de una página centrada a medida que aumenta la anchura de la ventana del navegador.

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

12. CENTRAR HORIZONTALMENTE

La solución consiste en asignar unos márgenes laterales automáticos al elemento que se desea centrar.

```
#contenedor {  
    width: 800px;  
    margin: 0 auto;  
    background-color: white;  
}
```

```
<body style="background-color: lightgray;">  
    <div id="contenedor">  
        <h1>Lorem ipsum dolor sit amet</h1>  
        Lorem ipsum dolor sit amet, consectetur...  
    </div>  
</body>
```

El valor 0 auto significa que los márgenes superior e inferior son iguales a 0 y los márgenes laterales toman un valor de auto. Cuando se asignan márgenes laterales automáticos a un elemento, los navegadores centran ese elemento respecto de su elemento padre.

12. CENTRAR HORIZONTALMENTE

Modificando ligeramente el código CSS anterior se puede conseguir un diseño dinámico o líquido (también llamado fluido) que se adapta a la anchura de la ventana del navegador y permanece siempre centrado.

Estableciendo la anchura del elemento contenedor mediante un porcentaje, su anchura se adapta de forma continua a la anchura de la ventana del navegador.

```
#contenedor {  
    width: 80%;  
    margin: 0 auto;  
    background-color: white;  
}
```



13. ALTURA/ANCHURA MÁXIMA

Cuando se diseña la estructura de una página web, se debe tomar la decisión de optar por un diseño de anchura fija o un diseño cuya anchura se adapta a la anchura de la ventana del navegador.

Sin embargo, la mayoría de las veces sería conveniente una solución intermedia: que la anchura de la página sea variable y se adapte a la anchura de la ventana del navegador, pero respetando ciertos límites. En otras palabras, que la anchura de la página no sea tan pequeña como para que no se puedan mostrar correctamente los contenidos y tampoco sea tan ancha como para que las líneas de texto no puedan leerse cómodamente.

CSS define cuatro propiedades que permiten limitar la anchura y altura mínima y máxima de cualquier elemento de la página. Las propiedades son ***max-width***, ***min-width***, ***max-height*** y ***min-height***, aunque las más importantes serán las de anchura, pues la altura de la página, casi en el 99,9% de los casos será mayor que la pantalla de visualización (piénsese en los dispositivos móviles).

13. ALTURA/ANCHURA MÁXIMA

De esta forma, para conseguir un diseño de anchura variable pero controlada, se podrían utilizar reglas CSS como la siguiente:

```
html, body { min-width: 500px; max-width: 900px; }
```

Las propiedades que definen la altura y anchura máxima y mínima se pueden aplicar a cualquier elemento, aunque solamente suelen utilizarse para estructurar la página. En general, las propiedades más utilizadas son ***max-width*** y ***min-width***, ya que no es muy habitual definir alturas máximas y mínimas.

13. ALTURA/ANCHURA MÁXIMA

Asimismo, se puede establecer una selección para una anchura máxima de LA PANTALLA DEL USUARIO, haciendo más sensible (“responsive”) el diseño, para lo cual habrá que incluir en el documento CSS varias secciones:

```
/* sección por defecto */  
html, body {min-width: 900px; max-width: 1010px;}  
  
/* sección “media queries” diseño para móviles o tablets hasta  
480px */  
@media screen and (max-width: 480px) {  
    html, body {min-width: 320px; max-width: 400px;}  
}
```

Esto lo logramos con las **media queries** que nos permite obtener información del dispositivo donde se muestra la página.

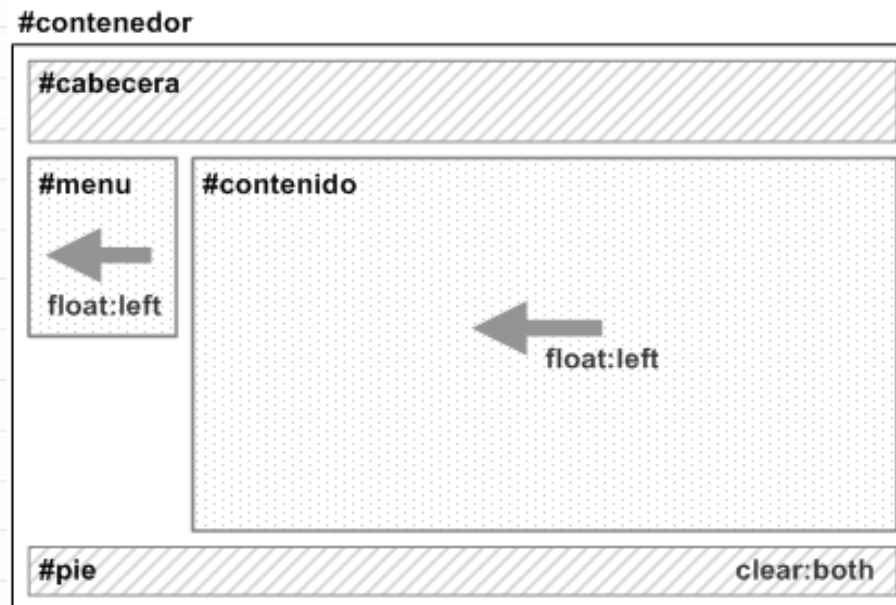
[CSS media queries](#)

14.1 EJEMPLOS LAYOUTS

Veremos algunos ejemplos de *layouts* aunque están simplificados. Es mejor usar las etiquetas relativas a la web semántica que permite HTML 5.

A. Diseño a dos columnas con cabecera y pie de página

La solución CSS se basa en el uso de la propiedad *float* para los elementos posicionados como el menú y los contenidos y el uso de la propiedad *clear* en el pie de página para evitar los solapamientos ocasionados por los elementos posicionados con *float*.



14.1 EJEMPLOS LAYOUTS

```
<style>
  #contenedor {
    width: 700px;
    margin: auto; }
  #cabecera {
    height: 20%; }
  #menu {
    float: left;
    width: 100px; }
  #contenido {
    float: left; /* tambien podria ser float:right; */
    width: 596px; /* hay que tener en cuenta border */
    height: 60%; }
  #pie {
    clear: both;
    height: 20%; }
</style>
<body>
  <div id="contenedor">
    <div id="cabecera">Cabecera</div>
    <div id="menu">Menu</div>
    <div id="contenido">Contenido</div>
    <div id="pie">Pie</div>
  </div>
</body>
```

Código HTML y CSS mínimos para definir la estructura de la página usando **<div>**

14.1 EJEMPLOS LAYOUTS

```
<style>
  html, body {
    width: 700px;
    margin: auto; }
  header {
    height: 20%; }
  nav {
    float: left;
    width: 15%; }
  section {
    float: left;
    width: 85%; }
  footer {
    clear: both; }
</style>
```

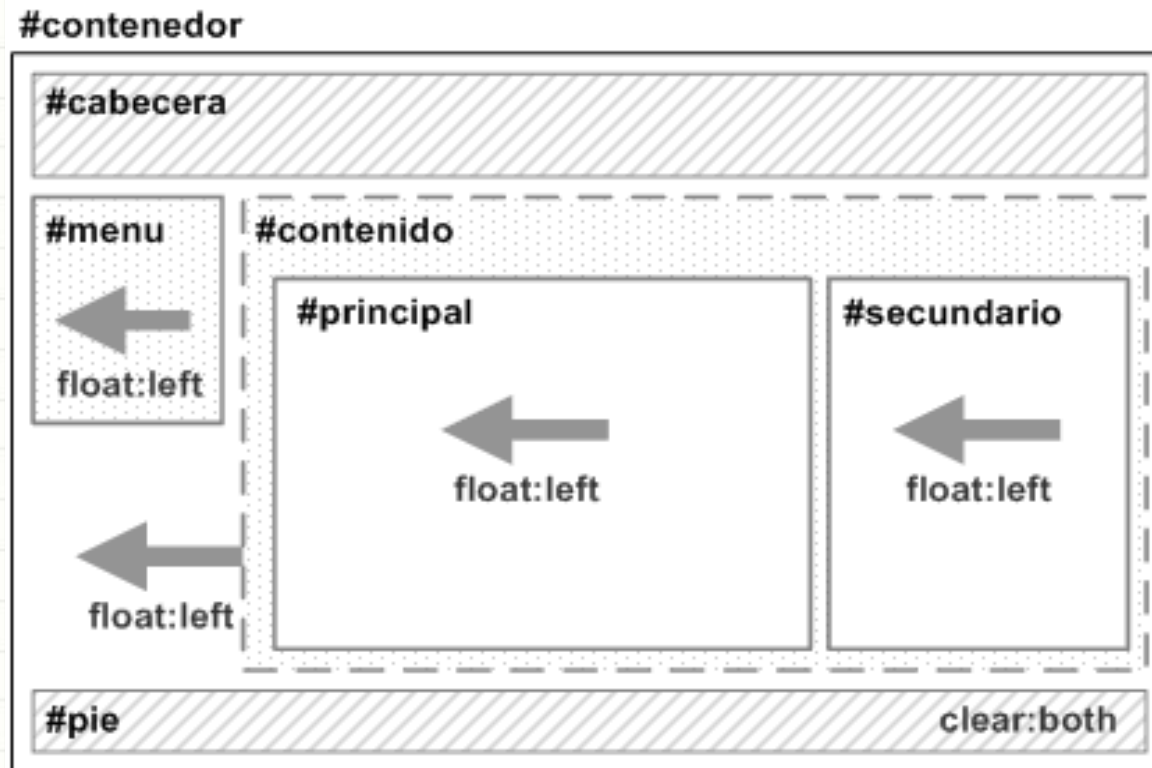
Código usando etiquetas específicas de HTML 5 y CSS

```
<body>
  <header>Cabecera</header>
  <nav>Menu</nav>
  <section>Contenido</section>
  <footer>Pie</footer>
</body>
```

14.2 EJEMPLOS LAYOUTS

B. Diseño a 3 columnas con cabecera y pie de página

Además del diseño a dos columnas, el diseño más utilizado es el de tres columnas con cabecera y pie de página. En este caso, los contenidos **<section>** se dividen en dos zonas diferenciadas: zona principal de contenidos **<article>** y zona lateral de contenidos auxiliares **<aside>**:



14.2 EJEMPLOS LAYOUTS

```
<style>
#contenedor { width: 700px; }
#menu {
  float: left;
  width: 15%; }
#contenido {
  float: left;
  width: 85%; }
#principal {
  float: left;
  width: 80%;}
#secundario {
  float: left;
  width: 20%;}
#pie { clear: both; }
</style>
<body>
  <div id="contenedor">
    <div id="cabecera"> cabecera </div>
    <div id="menu"> menu </div>
    <div id="contenido">
      <div id="principal"> principal </div>
      <div id="secundario"> secundario </div>
    </div>
    <div id="pie"> pie </div>
  </div>
</body>
```

Código HTML y CSS mínimos para definir la estructura de la página usando **<div>**, de anchura variable y que se adapte de forma dinámica a la ventana del navegador, sin aplicar ningún estilo adicional.

14.2 EJEMPLOS LAYOUTS

```
<style>
  html, body {
    width: 700px; }
  nav {
    float: left;
    width: 15%; }
  section {
    float: left;
    width: 85%; }
  article {
    float: left;
    width: 80%; }
  aside {
    float: left;
    width: 20%;}
  footer {
    clear: both; }
</style>
<body>
  <header>Cabecera</header>
  <nav>Menu</nav>
  <section>
    <article>Principal</article>
    <aside>Secundario</aside>
  </section>
  <footer>Pie</footer>
</body>
```

Si se hiciera uso de las etiquetas HTML5 header, nav, section, article, aside y footer, todo lo anterior quedaría como sigue.