

**CSS**



# **CSS. CASCADING STYLE SHEETS**

## ***HOJAS DE ESTILO EN CASCADA***

**UNIDAD 2**

# INTRODUCCIÓN

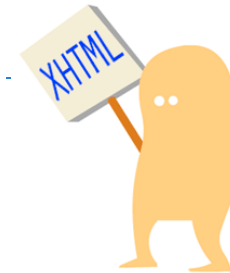
---

- ▶ CSS es un estándar del consorcio WWW o W3C, ampliamente reconocido y utilizado.
- ▶ CSS nos permite definir la presentación o estilo que le aplicaremos a:
  - ▶ Una web completa de una sola vez.
  - ▶ Un documento HTML o página determinada.
  - ▶ Un trozo concreto de una página.
  - ▶ Una etiqueta en concreto, llegando incluso a poder definir varios estilos diferentes para una sola etiqueta.
- ▶ CSS nos permite:
  - ▶ Ahorrar tiempo en el diseño de sitios web.
  - ▶ Conseguir efectos potentes, soportados por la mayoría de los navegadores.

# INTRODUCCIÓN

contenido

aspecto



## ► VENTAJAS DEL USO DE CSS

### ► Separar el diseño del contenido

- La principal ventaja del uso de CSS en el desarrollo de un sitio web, es que **permite separar el diseño** (la apariencia del sitio) **del contenido** (información del sitio).
- Por tanto, nos va a permitir **un desarrollo y mantenimiento más eficiente** de sitios web.
- Facilidad para alterar el aspecto de la página sin tocar el código HTML.
- Visitar <http://www.csszengarden.com/>

### ► Ahorro en la transferencia

- Si el código del estilo de la página se encuentra en un fichero externo haremos las **páginas del sitio web más ligeras**.
- La declaración de estilos se almacena en la caché del navegador y sólo se transfiere cuando se carga la primera página del sitio, por lo que las siguientes páginas se cargarán más rápido.

# INTRODUCCIÓN

---

## ► VENTAJAS DEL USO DE CSS

- Permite la adaptación de un sitio web a diferentes dispositivos en los que será visualizado (tablet, smartphone, portátil, pc de sobremesa, televisión,...).
- Cuando el servidor detecta el dispositivo cliente, éste aplica una serie de reglas de estilos para un mejor visionado.
- Esto se consigue actualmente con las Media Queries de CSS3.



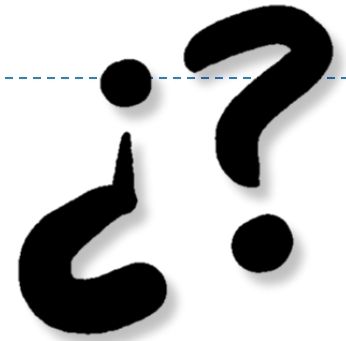
# **Aplicando estilos con CSS**

# Aplicando estilos con CSS

---

## ► ¿Dónde añadir CSS en un sitio web?

- En cualquier etiqueta.
- En la cabecera del documento HTML `<STYLE>`
- En un archivo CSS externo (.css) que será enlazado en la cabecera de cada página del sitio web, es decir en el `<head>`, a través de la etiqueta `<LINK>`



¡¡Vamos a verlo!!

# Estilo en una pequeña parte de la página

- ▶ Para definir estilos en secciones de una página se utiliza la etiqueta **<SPAN>** y el atributo **style**.
- ▶ Ejemplo:



# Estilo definido para una etiqueta

- Podemos hacer que toda una etiqueta muestre un estilo determinado:





# Estilo definido es una sección

- ▶ Con la etiqueta **<DIV>** podemos definir **secciones** dentro una página y aplicarle estilos con el atributo **style**.

```
ejemplo3CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>APRENDIENDO CSS</title>
5     <meta charset="utf-8">
6   </head>
7   <body>
8     <p>Estamos practicando con CSS</p>
9     <div style="color:#000099; font-weight:bold">
10       <h3>Esta sección de la página va en azul y negrita</h3>
11       <p>
12         Este párrafo está dentro del DIV, por tanto se aplican los estilos.
13       <br>
14       Es muy importante tabular el código para evitar cometer errores.
15     </p>
16   </div>
17 </body>
18 </html>
```

Separados por ;



# Estilo definido en una sección

---

- ▶ Como es lógico, podemos aplicar estilo en cualquier etiqueta estructural de HTML5 como son <header>, <section>, <article>, <aside>, <footer>,...

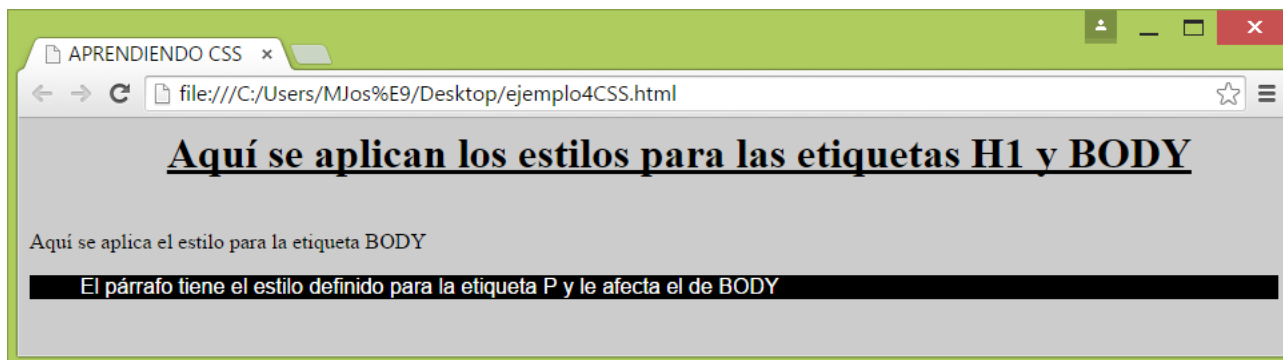
# Estilo para toda la página

---

- ▶ Podemos definir **en la cabecera** del documento estilos que serán aplicados a toda la página.
- ▶ Para ello utilizamos la **etiqueta** `<style>` dentro del `<head>` del documento html.
- ▶ Entre `<style>` y `</style>` se coloca el nombre de la etiqueta para la que queremos definir los estilos y entre llaves `{ }` especificamos el estilo.

```
<head>
  <title>APRENDIENDO CSS</title>
  <style type="text/css">
    <!--
      body{color:black;background-color:#CCCCCC;text-indent:1cm}
      h1{text-decoration:underline;text-align:center}
      p{font-family:arial,verdana;color:white;background-color:black}
    //-->
  </style>
</head>
```

```
ejemplo4CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>APRENDIENDO CSS</title>
5     <style type="text/css">
6       <!--
7         body{color:black;background-color:#CCCCCC;text-indent:1cm}
8         h1{text-decoration:underline;text-align:center}
9         p{font-family:arial,verdana;color:white;background-color:black}
10      //-->
11     </style>
12  </head>
13  <body>
14    <h1>Aquí se aplican los estilos para las etiquetas H1 y BODY</h1>
15    <br>
16    Aquí se aplica el estilo para la etiqueta BODY
17    <br>
18    <p>El párrafo tiene el estilo definido para la etiqueta P y le afecta el de BODY</p>
19  </body>
20 </html>
```



# Estilo para toda la página

- Analicemos el código CSS:

```
<style type="text/css">
```

```
<!--
```

```
body{
```

```
    color:black;
```

```
    background-color:#CCCCCC;
```

```
    text-indent: 1cm
```

```
}
```

```
h1{
```

```
    text-decoration:underline;
```

```
    text-align:center
```

```
}
```

```
p{
```

```
    font-family:arial,verdana;
```

```
    color:white;
```

```
    background-color:black
```

```
}
```

```
//-->
```

```
</style>
```

Color de fuente negro,  
Sombreado gris y  
Margen lateral de 1cm

Texto subrayado y centrado

Tipo de fuente arial o en su defecto verdana  
Color de fuente blanco y fondo negro

*Los comentarios HTML <!-- --> sirven para evitar que los navegadores antiguos que no soportan CSS impriman el código.*

# Estilo para todo el sitio web

---

- ▶ Para aplicar estilo a todo el sitio web a la vez, crearemos un **archivo externo** con las declaraciones de estilos, y enlazaremos este archivo con cada página del sitio web.
- ▶ De este modo, todas las páginas comparten una misma declaración de estilos y por tanto, si las cambiamos, cambiarán todas las páginas.
- ▶ Así estaremos **reduciendo el tamaño de cada página** del sitio web y será mucho más **fácil modificar** la apariencia del sitio web.



# Estilo para todo el sitio web

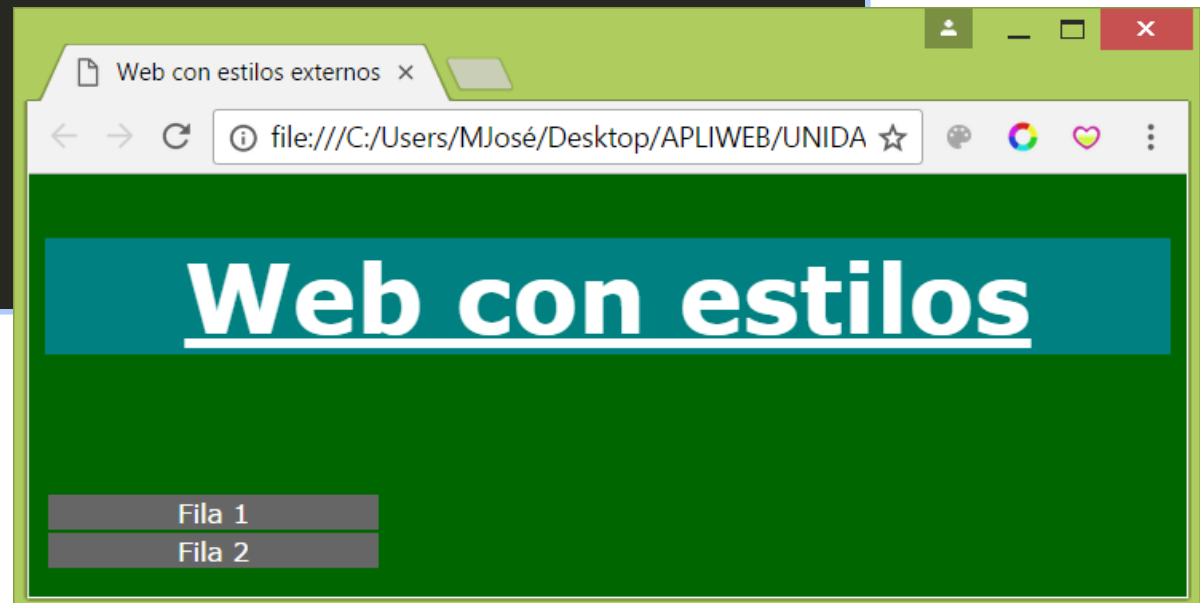
---

- ▶ Por tanto, tendremos que realizar dos acciones diferenciadas:
  - ▶ **Crear el fichero con la declaración de estilos**
    - ▶ Se tratará de un fichero de texto con extensión **.css** que contendrá **exclusivamente código CSS**.
    - ▶ Una hoja de estilo CSS está formada por **reglas de estilo**.
  - ▶ **Enlazar la página web con la hoja de estilos**
    - ▶ Para ello, necesitaremos la etiqueta **<link>** que incluiremos en la cabecera de cada página del sitio web con los atributos:
      - **rel="stylesheet"** indica que el enlace es con una hoja de estilos.
      - **type="text/css"** indica que se trata de un archivo de texto con código css.
      - **href="estilos.css"** indica el nombre del fichero de estilos.
    - ▶ También podremos utilizar la sentencia **@import url("estilos.css")**

```
body{
    background-color:#006600;
    font-family:arial;
    color:white;
}
p{
    font-size:12pt;
    font-family:arial,helvetica;
    font-weight: normal;
}
h1{
    font-size:36pt;
    font-family:verdana,arial;
    text-decoration: underline;
    text-align: center;
    background-color: teal;
}
table{
    width: 30%;
    border:0px;
    border-spacing: 2px;
}
td{
    font-size:10pt;
    font-family:verdana,arial;
    text-align:center;
    background-color:#666666;
}
```



```
ejemplo5CSS_link.html x
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Web con estilos externos</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="estilos.css">
  </head>
  <body>
    <h1>Web con estilos</h1>
    <br>
    <br>
    <table>
      <tr>
        <td>Fila 1</td>
      </tr>
      <tr>
        <td>Fila 2</td>
      </tr>
    </table>
  </body>
</html>
```



# Otra forma de importar estilos

- ▶ Podemos importar una declaración externa de estilos CSS con: **@import url("estilos.css")**
- ▶ Se debe incluir en la declaración de estilos global a una página, es decir, dentro de la etiqueta <style>.
- ▶ **IMPORTANTE**
  - ▶ La sentencia de importación del archivo CSS se debe escribir **en la 1ª línea** de la declaración de estilos:

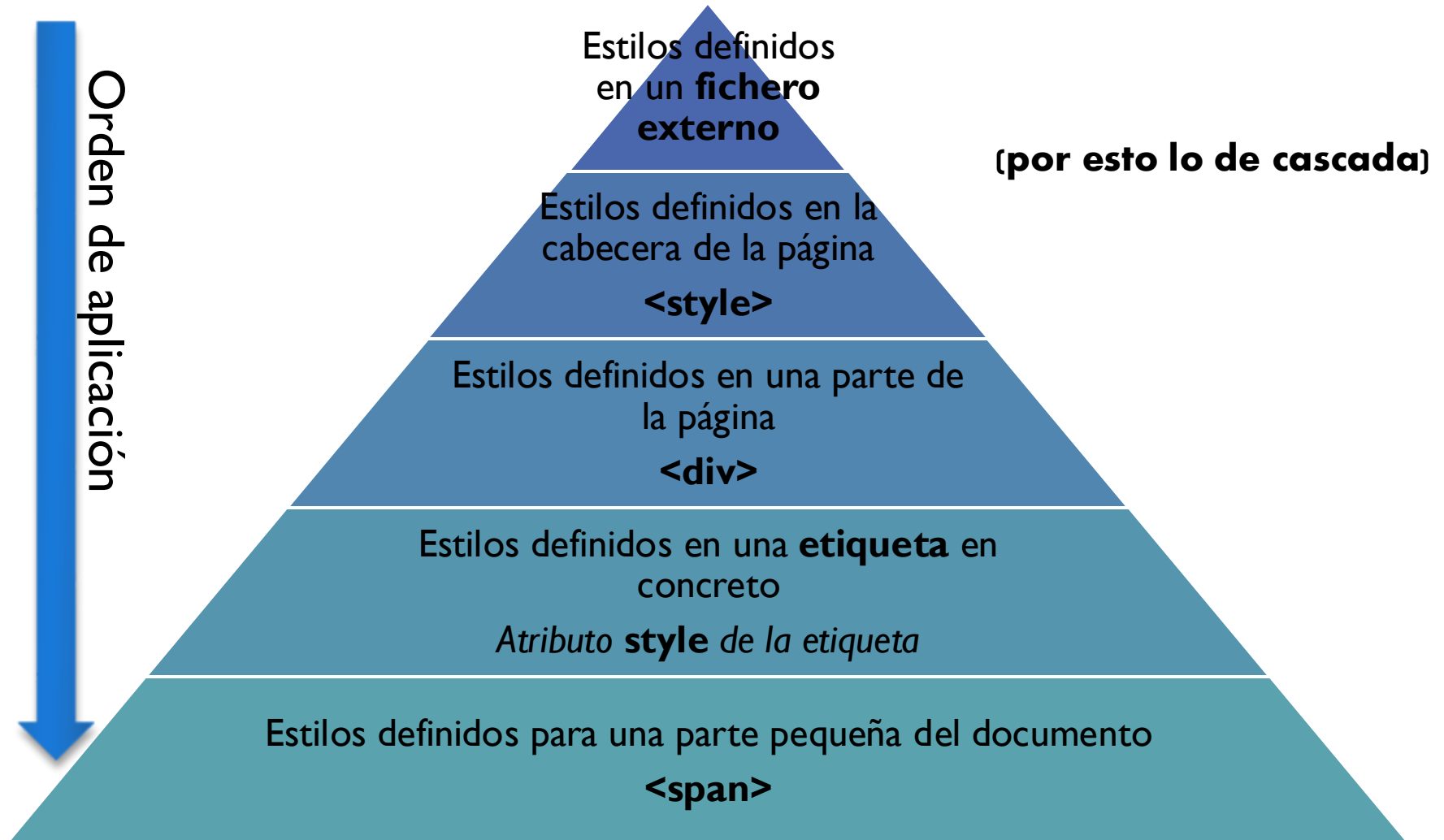
```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Web con estilos externos</title>
    <meta charset="utf-8">
    <style type="text/css">
      @import url("estilos.css");
    </style>
  </head>
```

# Reglas de importancia en los estilos

---

- ▶ Los estilos se heredan de una etiqueta a otra. Por ejemplo, los estilos de la etiqueta `<body>` afectarán a todas las etiquetas que estén dentro del cuerpo del documento.
- ▶ Puede ser que **más de una declaración de estilos afecte a una misma porción de código de la página.**
- ▶ Siempre se tiene en cuenta la declaración **más particular.**
- ▶ Atendiendo a los diferentes modos de definir estilos en una web, existe una **jerarquía de importancia para resolver conflictos** entre varias declaraciones de estilos distintas para una misma porción de página.

# Jerarquía de importancia ante conflictos



# SINTAXIS

---

## ► Para definir un estilo

- **atributo:valor; atributo:valor;...**

- ***Ejemplo:***

- font-size:10pt; text-decoration:underline; color:black;

- El último ; de la lista de atributos es opcional, pero para evitar posibles errores es mejor acostumbrarse a insertar siempre ;

## ► Para definir un estilo en una etiqueta

- **etiqueta{atributo:valor; atributo:valor;...}**

- ***Ejemplo:***

- h1{text-align:center; color:black;}

# Definir estilos utilizando clases

SELECTOR DE CLASE

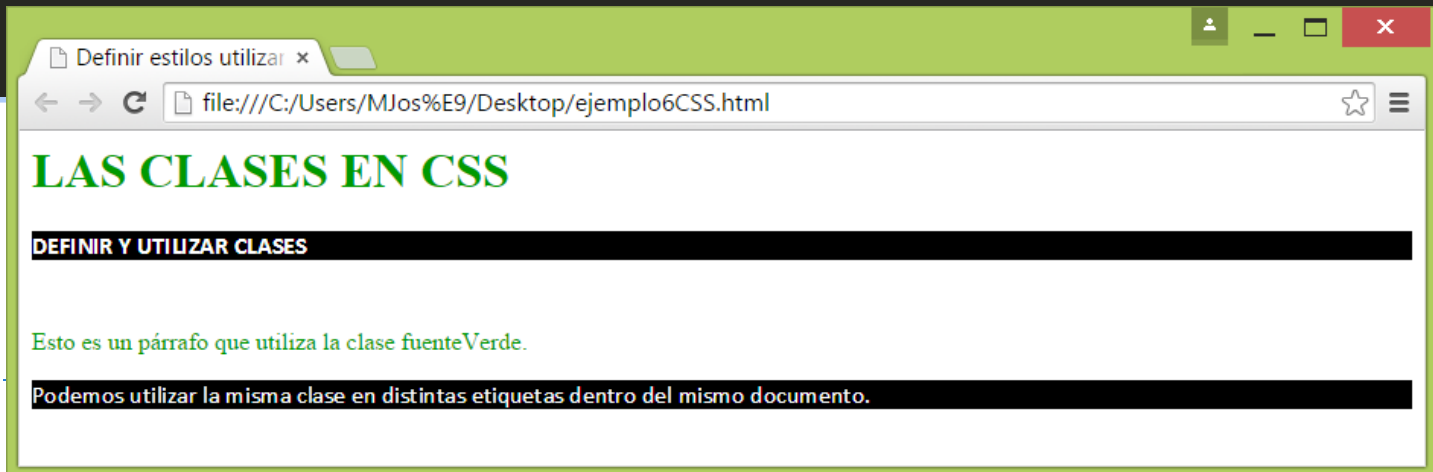
# Definir estilos utilizando **clases**

---

- ▶ Mediante las clases se pueden definir estilos abstractos, es decir, que no estén asociados directamente a una etiqueta HTML.
- ▶ Las clases nos van a servir **para definir estilos que se van a utilizar varias veces** en distintos tipos de etiquetas.
- ▶ Una clase se puede definir en la **cabecera** de la página dentro de la etiqueta **<style>** o en un **archivo externo .css <link>**.
- ▶ **Sintaxis:**
  - ▶ **.nombreDeLaClase{atributo: valor; atributo2:valor; ...}**
  - ▶ Las clases las podemos utilizar en cualquier etiqueta HTML a través del atributo **class**.
    - ▶ **<etiqueta class=«nombreDeLaClase nombreOtraClase»>**

# Definir estilos utilizando **clases**

```
ejemplo6CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <title>Definir estilos utilizando clases</title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     .fuenteVerde{color:#009900}
8     .fuenteResaltada{color:white; font-size:12pt; font-family:Calibri; background-color:black}
9   </style>
10 </head>
11 <body>
12   <h1 class=fuenteVerde>LAS CLASES EN CSS</h1>
13   <h1 class=fuenteResaltada>DEFINIR Y UTILIZAR CLASES</h1>
14   <br>
15   <p class=fuenteVerde>
16     Esto es un párrafo que utiliza la clase fuenteVerde.
17   </p>
18   <p class=fuenteResaltada>
19     Podemos utilizar la misma clase en distintas etiquetas dentro del mismo documento.
20   </p>
21 </body>
22 </html>
```





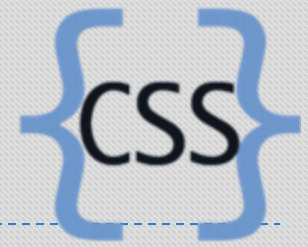
# Definir estilos utilizando **clases**

- ▶ También podríamos definir clases para etiquetas específicas de la siguiente forma:

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

```
<body>
  <h1 class=roja> Título rojo  </h1>
  <h1 class=verde> Título verde  </h1>
  <h2 class=azul> Título azul  </h1>
</body>
```

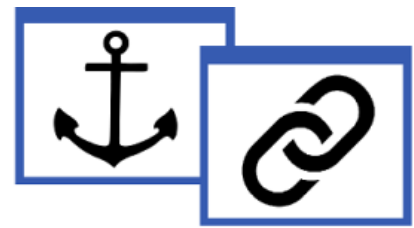
Es más eficiente definir clases independientes.



# EJERCICIO 1

- ▶ En un archivo externo define los siguientes estilos:
  - ▶ Para **body**, **td** y **p** color de fuente negro y fondo blanco.
  - ▶ Crea una **clase** llamada **inverso** con color de fuente blanco y fondo negro.
- ▶ Modifica el siguiente código para que se apliquen los estilos anteriores:

```
ejercicio1CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Ejercicio 1 sobre estilos</title>
5     <meta charset="utf-8">
6   </head>
7   <body>
8     <p> Este texto tiene que tener el estilo de la etiqueta P </p>
9     <P> Este texto tiene que tener el estilo definido en la clase </p>
10    <table>
11      <tr>
12        <td>Esta celda tiene el estilo definido en la clase</td>
13        <td>Esta celda tiene el estilo de la etiqueta TD </td>
14      </tr>
15    </table>
16  </body>
17 </html>
```



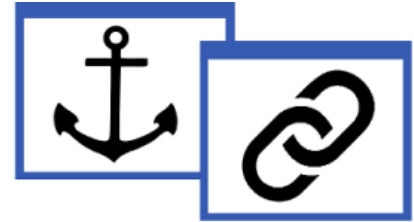
# Pseudo-clases

---

- ▶ **Pseudo-clases** que se pueden aplicar sobre los enlaces:
  - ▶ **Enlaces** que aún **no** han sido **visitados** por el usuario:
    - ▶ `a:link{color:red;}`
  - ▶ **Enlaces visitados** por el usuario:
    - ▶ `a:visited{color:green;}`
  - ▶ **Enlaces activos** (*Los enlaces están activos en el preciso momento en que se hace clic sobre ellos*)
    - ▶ `a:active{color:yellow;}`
  - ▶ **Enlaces hover** cuando el ratón se encuentra encima del enlace:
    - ▶ `a:hover{color:grey;}`

Las pseudo-clases `:link` y `:visited` solo se pueden aplicar sobre los enlaces. Sin embargo, `:active` y `:hover` se pueden aplicar a cualquier elemento.

# Pseudo-classes



```
ejemplo7CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <title>Definir estilos utilizando clases</title>
5   <meta charset="utf-8">
6   <style type="text/css">
7     a:link{text-decoration:none; color:blue;}
8     a:visited{text-decoration:none; color:red;}
9     a:active{text-decoration:none; color:yellow;}
10    a:hover{text-decoration:underline;color:grey;}
11  </style>
12 </head>
13 <body>
14   <a href="http://www.google.es" target="_blank">Ir a Google</a>
15 </body>
16 </html>
```

A screenshot of a web browser window. The title bar says "Definir estilos utilizar x". The address bar shows the file path: "file:///C:/Users/MJos%E9/Desktop/ejemplo7CSS.html". The page content displays a single blue link that says "Ir a Google".

# Pseudo-clases

---

- ▶ Algunas pseudo-clases nos van a ser muy útiles para seleccionar elementos del DOM a los que aplicarle estilo como, por ejemplo:
  - :root
    - Hace referencia al elemento raíz del documento que sería <html>
  - :nth-child
    - Permite seleccionar un hijo determinado.
  - :nth-last-child
    - Permite seleccionar elementos de una lista de elementos hermanos, contando hacia atrás desde el final de la lista.
  - :first-child
    - Para seleccionar el primer hijo.
  - :last-child
    - Para seleccionar el último hijo.

[Ver más pseudo-clases de CSS3](#)

# TÉCNICAS MODERNAS DE CSS

---

- Definir la paleta de colores con la pseudo-clase :root que hace referencia al elemento raíz del documento que sería <html>

```
1  :root{
2      /* Paleta de colores */
3      --color-green:  hsl(120, 100%, 80%);
4      --color-lighgray:  #eee;
5      --color-dark:  #222;
6      --color-black:  #000;
7
8      /* Estilos del tema */
9      --theme-primary: var(--color-green);
10     --theme-secondary: var(--color-lighgray);
11     --theme-background: var(--color-dark);
12     --theme-text: var(--color-lighgray);
```

# Definir estilos utilizando identificadores

SELECTOR DE IDENTIFICADOR

# Definir estilos utilizando identificadores

---

- ▶ 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 la clase.
- ▶ En CSS las clases se pueden usar en uno o varios elementos (etiquetas).
- ▶ Sin embargo, y ésta es la diferencia, **los identificadores sólo 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 «coger» ese objeto de estilo.



# SELECTOR DE IDENTIFICADOR

---

- ▶ La **sintaxis** utilizada para declarar un selector basado en identificador es prácticamente idéntica a la utilizada para definir un selector de clase.
- ▶ **Sintaxis:**
  - ▶ `#nombreIdentificador{atributo:valor; atributo:valor}`
- ▶ La diferencia existente entre un selector de clase y un selector de identificador es que este último se crea para ser utilizado **una sola vez**; en cambio los selectores de clase son creados para ser utilizados **varias veces** en un mismo sitio web.

# SELECTOR DE IDENTIFICADOR

## IDENTIFICADORES

▶ `<DIV id=«slogan»>`

...

▶ `</DIV>`

ÚNICO

▶ `#slogan{`  
    `color:#333333;`  
    `font-size:14px;`  
`}`

## CLASES

▶ `<DIV class=«destacado»>`

...

▶ `</DIV>`

PUEDE  
UTILIZARSE  
VARIAS  
VECES

▶ `.destacado{`  
    `color:red;`  
    `font-weight:bold;`  
`}`

# SELECTOR DE IDENTIFICADOR

---

## ▶ AVISO IMPORTANTE

- ▶ Realmente, en muchos navegadores actuales, si se usa un identificador en varios elementos, éstos se interpretan y el resultado es el mismo que si utilizáramos clases.
- ▶ Sin embargo, «**son buenas prácticas**» de diseñador usar los identificadores y las clases en su momento adecuado.
- ▶ De esta forma seguimos un estándar de uso y como diseñadores web, podemos beneficiarnos de las ventajas de los identificadores.
- ▶ Validando nuestros diseños CSS en el W3C nos aseguramos no haber duplicado por error identificadores.

# SELECTOR DE IDENTIFICADOR

- ▶ Ejemplo de uso de identificadores
- ▶ Identificar cada uno de los elementos principales del sitio web:

```
<header id="cabecera">
  <picture>
    
  </picture>
  <!-- Menú principal -->
  <nav id="menu-principal">
    <label for="menu-hamburger">
      <span>&#9776;</span>
    </label>
    <input type="checkbox" id="menu-hamburger"/> <!--Para la hamburguesa-->
    <ul id="menu">
      <li><a href="#">TODOS</a>
      <li><a href="#">NUEVA COLECCIÓN</a>
      <li><a href="#">NOVIA</a>
      <li><a href="#">OUTLET</a>
    </ul>
  </nav>
  <!--Bloque de enlaces con iconos -->
  <nav id="buscador">
    <ul>
      <li><a href="#"><i class="fa fa-search" aria-hidden="true"></i></a></li>
    </ul>
  </nav>
  <nav id="compra">
    <ul>
      <li><a href="#"><i class="fa fa-shopping-bag" aria-hidden="true"></i></a></li>
    </ul>
  </nav>
</header>
```

# SELECTOR DE IDENTIFICADOR

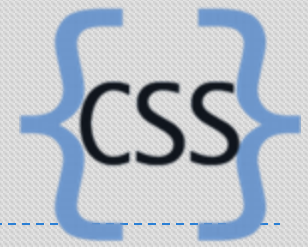
## ► Otro ejemplo de uso:

- CSS permite utilizar los identificadores como «**marcadores**» y utilizarlos en los enlaces de la siguiente forma:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Los selectores tipo identificador</title>
  <meta charset="utf-8">
  <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>
</html>
```

Enlace al elemento con el identificador «cabecera»



## EJERCICIO 2

---

- ▶ Crea un fichero HTML y define un selector de clase y un selector de identificador.
- ▶ Prueba a usar **id** y **class** en uno o varios elementos HTML y comprueba su efecto.
  - ▶ ¿Qué ocurre si no se cumple la especificación y se repite un identificador en varios elementos?
  - ▶ ¿Cómo responde el navegador?
  - ▶ Y en otro navegador, ¿responde igual?
  - ▶ Valida el código CSS en la web del w3c y comprueba si considera un error el hecho de que haya un identificador duplicado.

# Agrupación y anidamiento de selectores

# AGRUPACIÓN Y ANIDAMIENTO

---

- ▶ Los selectores se pueden **agrupar** y **anidar** para conseguir estilos CSS **más concretos y definidos** y para tener un fichero CSS **más optimizado y fácil de entender** por el equipo de desarrollo.
- ▶ Vamos a ver la diferencia entre **Agrupación** y **Anidamiento** de selectores.



# AGRUPAMIENTOS

---

- ▶ Los agrupamientos nos permiten **aplicar el mismo estilo a un conjunto de selectores** (etiqueta, clase o identificador) a la vez.
- ▶ Cualquier selector se puede agrupar siguiendo la siguiente **sintaxis**:
  - ▶ Selector1, Selector2, ...{atributo:valor; atributo:valor;...}



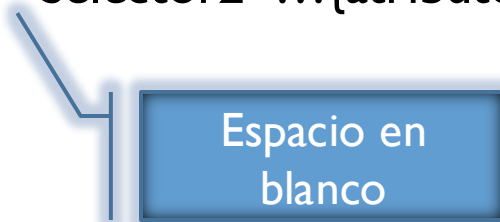
Separados por comas

- ▶ **Ejemplos:**
  - ▶ h1, p, h3 {font-family:arial;}
  - ▶ .mensaje, .subtitulo, .enfasis {color:#0000FF;}
  - ▶ #cabecera, #piepagina {color:#FF0000; font-family:verdana;}
  - ▶ .resaltado, h2, #tabla1 {text-decoration:underline;}

# ANIDAMIENTOS

---

- ▶ 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 a su alrededor.
- ▶ **Sintaxis:**
  - ▶ Selector1 Selector2 ...{atributo:valor; atributo:valor;...}



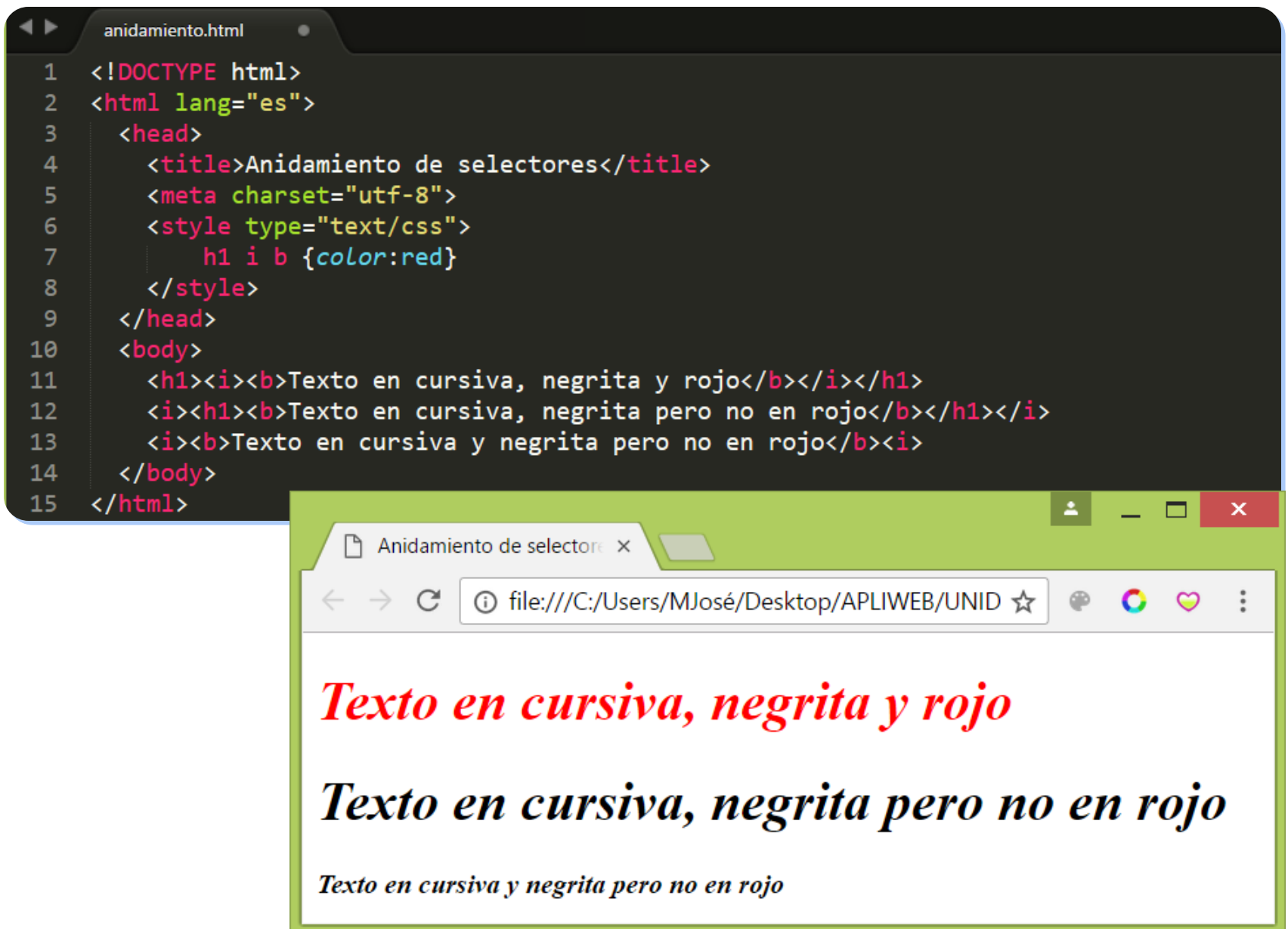
# ANIDAMIENTOS

## ► Ejemplo:

### ► h1 i b {color:red}

- Así se verá en color rojo un texto en negrita <b> siempre y cuando esté incluido dentro de una etiqueta de título <h1> y en cursiva <i> (aunque entre medias haya otras etiquetas pero en el mismo orden).

```
anidamiento.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Anidamiento de selectores</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       h1 i b {color:red}
8     </style>
9   </head>
10  <body>
11    <h1><i><b>Texto en cursiva, negrita y rojo</b></i></h1>
12    <i><h1><b>Texto en cursiva, negrita pero no en rojo</b></h1></i>
13    <i><b>Texto en cursiva y negrita pero no en rojo</b><i>
14  </body>
15 </html>
```




# ANIDAMIENTOS

---

- ▶ En principio no hay límite en el número de anidamientos que se pueden hacer, sin embargo **no es recomendable un anidamiento de más de 4 o 5 niveles** por motivos de complejidad a la hora de interpretar el CSS, tanto por el navegador como por los diseñadores.
- ▶ En los anidamientos se pueden utilizar etiquetas, clases e identificadores o una combinación de los tres.
- ▶ Además los **anidamientos** se pueden **agrupar** como muestra el siguiente ejemplo:
  - ▶ `h1 i b, h1 a {color:red;}`

# ANIDAMIENTOS

- ▶ **Anidamiento con selectores hijo** (descendant selector)
  - ▶ El anidamiento se comporta igual tanto si las etiquetas están consecutivas como si hay etiquetas intermedias.



The image shows a code editor on the left and a web browser on the right. The code editor displays the following HTML code:

```
1 <html lang="es">
2 <head>
3   <title>Anidamiento de selectores</title>
4   <meta charset="utf-8">
5   <style type="text/css">
6     h1 b {color:red}
7   </style>
8 </head>
9 <body>
10   <h1><i><b>Texto en cursiva, negrita y rojo</b></i></h1>
11   <h1><b>Texto en negrita y en rojo</b></h1>
12 </body>
13 </html>
```

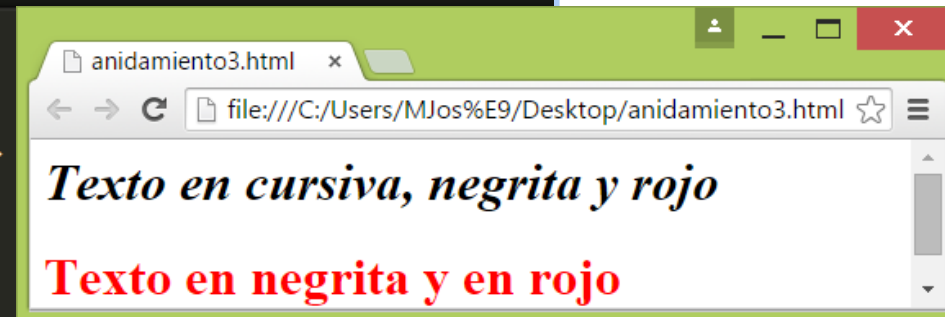
The web browser on the right shows the rendered output. The first line of text is *Texto en cursiva, negrita y rojo* (italic, bold, red). The second line of text is **Texto en negrita y en rojo** (bold, red).

# ANIDAMIENTOS

- ▶ Si queremos restringir que las etiquetas estén seguidas unas de otras, se tiene que utilizar la siguiente sintaxis:
  - ▶ **selectorX > selectorY {atributo:valor; ...}**



```
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Anidamiento de selectores</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       h1>b {color:red}
8     </style>
9   </head>
10  <body>
11    <h1><i><b>Texto en cursiva, negrita y rojo</b></i></h1>
12    <h1><b>Texto en negrita y en rojo</b></h1>
13  </body>
14 </html>
```



# ANIDAMIENTOS

## ► Otro ejemplo:

```
anidamiento4.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Anidamiento de selectores</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       h1>b>i {color:red}
8     </style>
9   </head>
10  <body>
11    <h1><i><b>Texto h1 en cursiva, negrita y negro</b></i></h1>
12    <h1><b><i><u>Texto h1 en negrita, cursiva, rojo y subrayado </u></i></b></h1>
13  </body>
14 </html>
```

En este ejemplo, ¿qué texto aparecerá formateado en color rojo?



# ANIDAMIENTOS

```
anidamiento4.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Anidamiento de selectores</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       h1>b>i {color:red}
8     </style>
9   </head>
10  <body>
11    <h1><i><b>Texto h1 en cursiva, negrita y negro</b></i></h1>
12    <h1><b><i><u>Texto h1 en negrita, cursiva, rojo y subrayado </u></i></b></h1>
13  </body>
14 </html>
```



# ANIDAMIENTOS

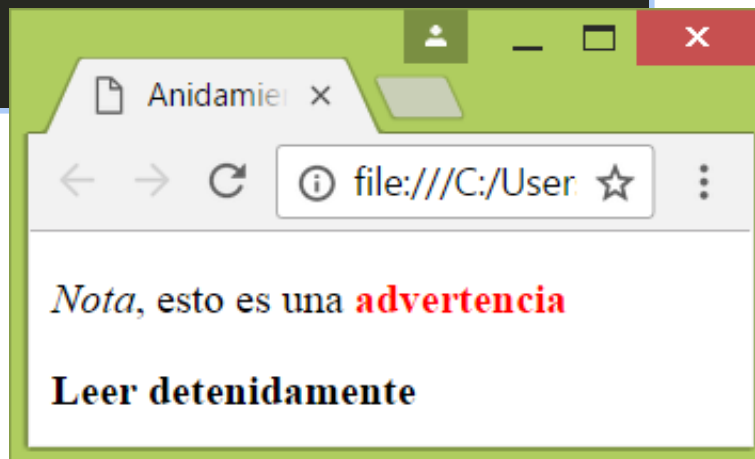
---

- ▶ **Anidamiento de selectores adyacentes hermanos** (Adjacent Sibling Selector)
  - ▶ El selector de hermanos adyacentes selecciona todos los elementos que son los hermanos adyacentes de un elemento especificado (en primer lugar). Los elementos hermanos deben tener el mismo elemento padre, y "adyacente" significa "inmediatamente después".
  - ▶ La sintaxis es la siguiente (para dos selectores):
    - ▶ `SelectorX+SelectorY {atributo:valor; atributo;valor;...}`
  - ▶ En el siguiente ejemplo, solo se aplicará el estilo sobre la etiqueta `<b>` si hay una etiqueta `<i>` adyacente es decir, en el mismo nivel de anidamiento y justo encima de dicha etiqueta `<b>`:
    - ▶ `i+b {color:red}`

# ANIDAMIENTOS

```
anidamiento5.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Anidamiento de selectores</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       i+b {color:red}
8     </style>
9   </head>
10  <body>
11    <p><i>Nota</i>, esto es una <b>advertencia</b></p>
12    <b>Leer detenidamente</b>
13  </body>
14 </html>
```

<b> está adyacente a <i> y en el mismo nivel de anidamiento

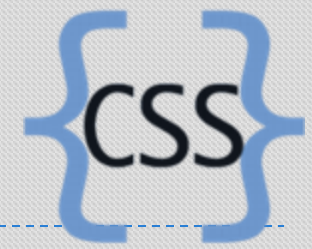


Todos los anidamientos se pueden hacer con etiquetas, clases, identificadores o combinaciones

# ANIDAMIENTOS

---

- ▶ Anidamientos de selectores hermanos generales General Sibling Selector
  - ▶ El combinador ~ separa dos selectores y selecciona el segundo elemento sólo si está precedido por el primero y ambos comparten un padre común.
  - ▶ Sintaxis:
    - ▶ elemento ~ elemento { atributo:valor; atributo:valor;...}
  - ▶ Ejemplo:
    - ▶ `div ~ p {background-color: yellow;}`
      - Aplica el color de fondo a todos los párrafos que tenga como **hermano mayor** (que esté posicionado en el código **antes pero no justo encima**) un contenedor <div>.
  - ▶ Ir al siguiente enlace para entenderlo mejor:  
[https://www.w3schools.com/css/css\\_combinators.asp](https://www.w3schools.com/css/css_combinators.asp)



## EJERCICIO 3

- Interpreta qué hace el siguiente código. ¿De qué color y tamaño aparecerán los elementos de la lista? Una vez analizado el código pruébalo en el navegador.

```
ejercicio3CSS.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Agrupamiento y anidamiento CSS</title>
5     <meta charset="utf-8">
6     <style>
7       i+b {color:red}
8       .nivel1 {color:green}
9       .nivel2 {color:blue}
10      ul ul .nivel2 {font-size:x-small}
11      ul ul li {font-size:small; color:purple}
12    </style>
13  </head>
14  <body>
15    <h1><p><i>Título</i>: <b>Anidamiento y agrupamiento</b> de selectores </p></h1>
16    <ul>
17      <li class="nivel1"> Agrupamiento</li>
18      <li class="nivel2">Anidamiento</li>
19      <ul>
20        <li class="nivel2"> Anidamiento de hijos</li>
21        <li class="nivel2"> Anidamiento de adyacentes</li>
22        <li>Anidamiento común</li>
23      </ul>
24    </ul>
25  </body>
26 </html>
```

# EJERCICIO 3

---

## ► Solución:



# RESUMEN

---

## ▶ AGRUPAMIENTOS

- ▶ Aplicar el mismo estilo a varias etiquetas, clases y/o identificadores a la vez.
  - ▶ `.resaltado, h2, #tabla1 {text-decoration:underline;}`

## ▶ ANIDAMIENTOS

- ▶ Los selectores tienen que aparecer en el orden establecido para aplicar el estilo.

### ▶ Anidamientos con selectores hijo

- ▶ Puede haber otras etiquetas entre medias
  - `.h1 i b {color:red}`
- ▶ Obligar a que las etiquetas estén seguidas
  - `h1>b {color:red}`

### ▶ Anidamientos con selectores adyacentes en el mismo nivel de anidamiento (hermanos)

- ▶ `i+b {color:red}`
  - `<p><i>Texto</i>para<b>imprimir</b><span>y algo</span><b>más</b></p>`
- ▶ `div~p {color:red}`
  - `<div>Texto</div><p>Texto1</p><code>Código</code><p>Texto2</p>`

### ▶ Los anidamientos se pueden agrupar:

- ▶ `h1 i b, h1 a {color:red}`

# TÉCNICAS MODERNAS DE CSS

---

## ► Nesting de SASS de forma nativa

- El **nesting** en CSS (o **anidamiento**) es una característica que permite escribir reglas de estilo dentro de otras reglas de estilo, lo que ayuda a organizar el código de manera más jerárquica y lógica.
- Es similar al anidamiento que se puede hacer en preprocesadores como SASS o LESS, pero aplicado directamente en CSS.
- El único inconveniente es que al ser algo novedoso, no es soportado por todos los navegadores actuales.



# TÉCNICAS MODERNAS DE CSS

---

## ► Nesting de SASS de forma nativa

```
.navbar {  
  background-color: #333;  
}  
  
.navbar ul {  
  list-style-type: none;  
}  
  
.navbar ul li {  
  display: inline-block;  
}
```

```
.navbar {  
  background-color: #333;  
  
  ul {  
    list-style-type: none;  
  
    li {  
      display: inline-block;  
    }  
  }  
}
```

# TÉCNICAS MODERNAS DE CSS

---

## ► :has

- Según la condición que tenga el hijo podremos cambiar el selector padre.

```
.learnings:has(#list:checked) .learnings-list{  
    grid-template-columns: max-content;  
}
```

Aplica el estilo a la clase .learnings-list si tiene un padre con la clase .learnings con un elemento con id #list (input) que esté checked.

# TÉCNICAS MODERNAS DE CSS

## ► Nesting de SASS de forma nativa

```
.learnings-selector{
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 1rem;
  margin-bottom: 2rem;
  & label{ /* NESTING: Equivale a .learnings-selector label. Nos permite escribir en SAAS en nativo*/
    cursor: pointer;
    &:hover {
      filter: invert(1); /*Invierte el color; funciona muy bien de blanco a negro*/
    }
    &:has(+input:checked) { /*Cuando la etiqueta label tengo un elemento input adyacente que está checked.
      El estilo se aplica en el label cuando está hover*/
      filter: invert(0.5);
    }
  }
}
```

Es necesario instalar la extensión [CSS Nesting Syntax Highlighting](#) para que Visual Estudio no marque el nesting como un error.

Ver ejemplo de nesting en Moodle.

# TÉCNICAS MODERNAS DE CSS

---

- :is
- :not
- :focus
- :placeholder-shown

```
textarea:is(:focus,:not(:placeholder-shown)){  
  height: 200px;  
  outline: none; /*No se debe quitar por accesibilidad*/  
}
```

# BUENAS PRÁCTICAS AL ESCRIBIR CSS

# BUENAS PRÁCTICAS

---

- ▶ El gran número de reglas que puede tener el CSS de un sitio web profesional, 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 suelen tener en cuenta.
- ▶ Seguir estas prácticas no solo ayuda a un desarrollo propio más claro, sino que también ayuda a entender mejor el desarrollo de otros autores.

# BUENAS PRÁCTICAS

---

- ▶ Los **selectores** se nombran en **minúsculas**, nunca empezando por números o caracteres especiales.
  - ▶ La especificación CSS no lo permite para nombres de clases e identificadores.
  - ▶ CSS no distingue entre mayúsculas y minúsculas salvo en los selectores de clase e identificadores, así que **para evitar errores lo escribimos todo en minúscula**.
  - ▶ Aunque en las últimas versiones se permite usar «\_» en los nombres, en su lugar es preferible utilizar «-», ya que no todos los navegadores soportan «\_»
    - mi\_clase → peor
    - mi-clase → mucho mejor

# BUENAS PRÁCTICAS

---

- ▶ El nombre de los selectores debe ser específico y claro, para que tenga una mayor capacidad expresiva.
  - ▶ Por ejemplo, si voy a definir un estilo para el nivel I de una lista un nombre apropiado sería:
    - ▶ `.lista-nivelI{color:green}`
- ▶ El nombre de las clases e identificadores no debe describir una característica visual, como color, tamaño o posición.
  - ▶ Por ejemplo:
    - ▶ `.rojo{color:red}`
    - ▶ Si por cualquier motivo se desea cambiar el valor del atributo color por otro distinto, se debería cambiar también el nombre a la clase y habría que buscar dentro de todas las páginas del sitio web las referencias a la clase rojo para cambiar el nombre.



# BUENAS PRÁCTICAS

---

- ▶ Del mismo modo, no se deben asignar nombres a las clases e identificadores según su localización:
  - ▶ Por ejemplo: `.menu-izquierda{...}` no sería del todo adecuado. Sería más correcto, por ejemplo `.menu-navegacion{...}`
- ▶ Documentar el código CSS
  - ▶ `/* Esto es un comentario en CSS */`
  - ▶ Es importantísimo documentar el código fuente para que otros desarrolladores y nosotros mismos, no tengamos problemas en el futuro interpretando el código.
- ▶ Separar las palabras mediante guiones.
  - ▶ `.menu-navegacion{...}`

# BUENAS PRÁCTICAS

---

- ▶ No hacer uso excesivo de las clases.
  - ▶ Los selectores de clase se suelen dejar para las partes más relevantes de la estructura.
- ▶ Validar el código CSS haciendo uso del [validador gratuito de W3C](#).
- ▶ 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 agruparlas unas debajo de otras.
    - ▶ [...]
    - ▶ `table{border:double}`
    - ▶ `table.miembros{border:solid}`
    - ▶ `table.empleados{border:grove}`
    - ▶ [...]

# BUENAS PRÁCTICAS

---

- ▶ Definir al principio las reglas de estilo de etiquetas y delimitarlas por comentarios:

```
/*Etiquetas HTML*/
```

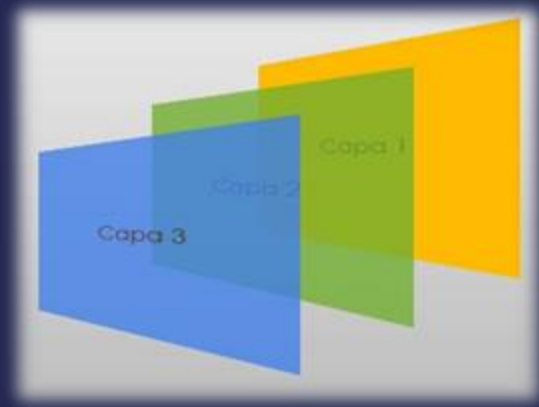
```
html {font-family:arial, verdana; font-size: 13px}
```

```
h2{color:green}
```

```
/*FIN Etiquetas HTML*/
```

- ▶ Definir las reglas de estilo de etiquetas siguiendo un orden lógico: por ejemplo `body{} h1{}` sería mejor que `h1{} body{}`.
- ▶ Estructurar visualmente el documento CSS utilizando tabulaciones.

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



# CAPAS FLOTANTES DE CSS

# CAPAS FLOTANTES

- ▶ La etiqueta **<div>** nos va a permitir crear capas en un documento Web.
- ▶ Una capa es una **división** dentro de una página que tiene un comportamiento independiente en el navegador.
- ▶ Ejemplo de creación de capa:

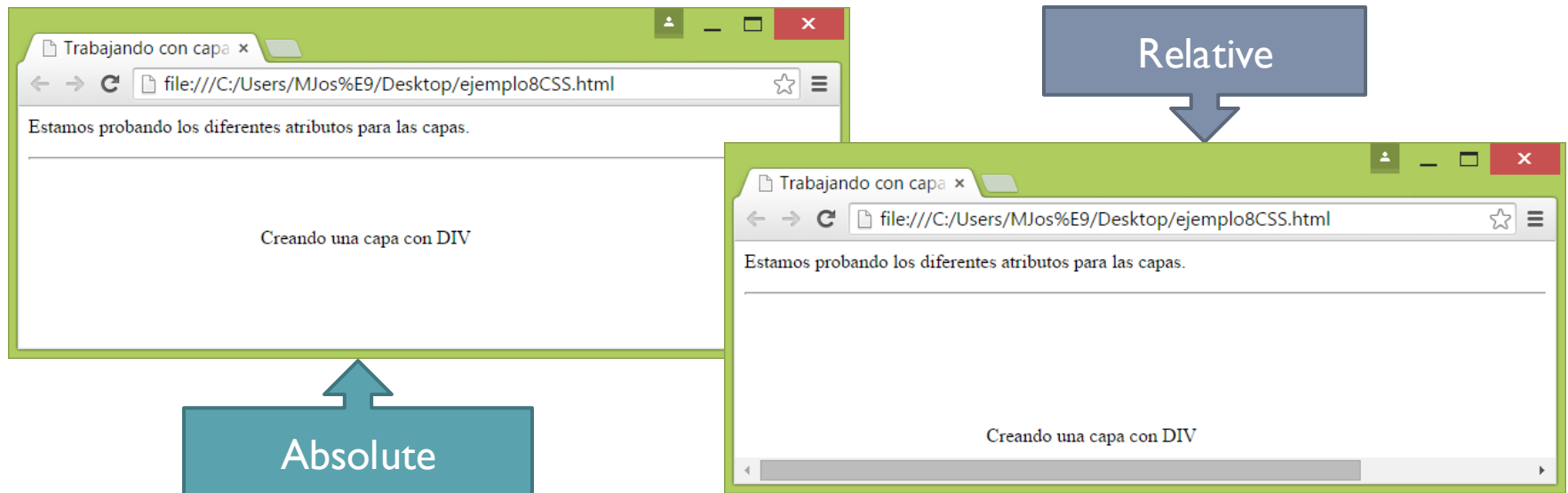
```
ejemplo8CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <title>Trabajando con capas</title>
5     <meta charset="utf-8">
6 </head>
7 <body>
8     <p> Estamos probando los diferentes atributos para las capas. </p>
9     <hr>
10    <div id="capa1" style="position:absolute; left:200px; top:100px;">
11        Creando una capa con DIV
12    </div>
13 </body>
14 </html>
```

# LAS CAPAS

## ► Atributos para capas

### ► Atributo position

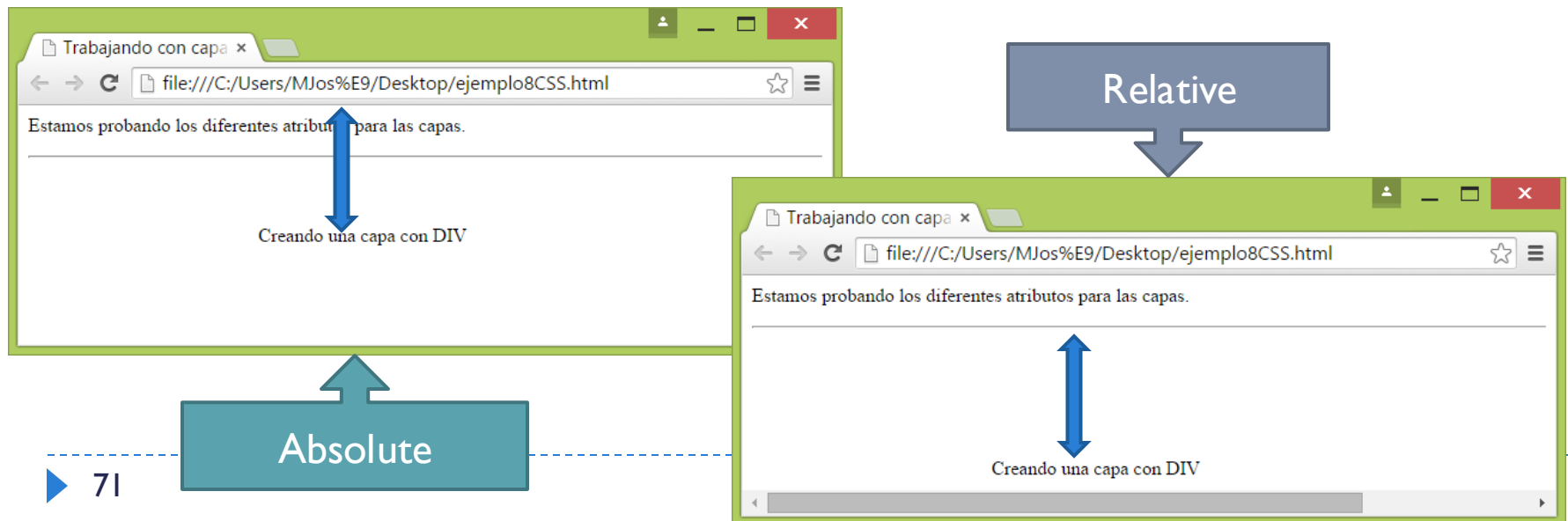
- Indica el posicionamiento de la capa. Puede tener dos valores:
  - ☐ **Relative**: indica que la posición de la capa es relativa a la posición donde se inserte la capa dentro de la página.
  - ☐ **Absolute**: indica que la posición de la capa se calcula con respecto al punto superior izquierdo de la página.



# LAS CAPAS. ATRIBUTOS

## ► **Atributo top**

- Indica la distancia en vertical donde se colocará la capa.
- Si **position:absolute**, **top** indica la distancia del borde superior de la capa con respecto al borde superior de la página.
- Si **position:relative**, **top** indica la distancia del borde superior de la capa con respecto a la posición de la capa dentro del documento.



# LAS CAPAS. ATRIBUTOS

---

## ▶ **Atributo width**

- ▶ Sirve para especificar la **anchura** de la capa.

## ▶ **Atributo height**

- ▶ Sirve para especificar la **altura** de la capa.

## ▶ Estos dos atributos pueden tomar los siguientes valores:

- ▶ auto | inherit | longitud | porcentaje
  - **auto**: valor por defecto, indica que el navegador tiene que calcular la anchura/altura del elemento teniendo en cuenta su contenido y el sitio disponible en la página web.
  - **inherit**: indica que la anchura/altura del elemento se hereda de su elemento padre.
  - **unidad de medida**: em, rem, ex, px, in, cm, mm, pt, pc, etc.
  - **porcentaje %**: se calcula a partir de la anchura/altura de su elemento padre.



# UNIDADES DE MEDIDA

---

## ► Unidades de Medida Absolutas o Fijas

- Permiten determinar el tamaño de forma concreta.
  - Píxeles (**px**), Pulgadas (**in**), Puntos (**pt**), Centímetros (**cm**), Milímetros (**mm**), Picas (**pc**).

## ► Unidades de Medida Relativas

- Permiten determinar el tamaño **en función de** otro tamaño.
  - % → relativo al tamaño del contenedor donde se encuentre el elemento. Por ejemplo: si especificamos un ancho del 50% para un elemento, su tamaño será el 50% del tamaño que tenga el contenedor donde se encuentre dicho elemento.
  - **em** → relativo al tamaño del texto del contenedor donde se encuentra.
  - **rem** → relativo al tamaño del texto del contenedor raíz.
  - **vw** (viewport width) → relativo a la anchura de la pantalla del dispositivo.
  - **vh** (viewport height) → relativo a la altura de la pantalla del dispositivo.

# UNIDADES DE MEDIDA PARA TEXTOS

---

## ► Tamaños fijos de fuente

- De forma tradicional se han utilizado **px** para establecer el tamaño de la fuente.
- Los navegadores web suelen tener un **valor por defecto de 16px** para la **fuentes**.
- Pero esto puede ocasionar algunos problemas:
  - No todos los navegadores pintan igual el px.
  - No es un valor escalable.
    - Para que la fuente sea responsive o adaptativa tendríamos que crear media queries para controlar todas las posibles situaciones.
    - Además de para establecer el tamaño de fuente, utilizamos las unidades de medida para muchas otras cosas así que mejor utilizar unidades de medida relativas.



# UNIDADES DE MEDIDA PARA TEXTOS

---

## ► Tamaños escalables de fuente

- Utilizar **em** para establecer el tamaño de la fuente.
  - Es una unidad relativa a su elemento padre.
  - Supongamos el siguiente CSS:

```
body{  
    font-size: 10pt;  
}  
div {  
    font-size: 1.5em;  
}  
strong {  
    font-size: 2em;  
}
```



# UNIDADES DE MEDIDA PARA TEXTOS

---

## ► Tamaños escalables de fuente

- Utilizar **em** para establecer el tamaño de la fuente.
  - Si el código HTML es el siguiente, ¿cuál es el tamaño de fuente en cada caso?

```
<!DOCTYPE html>
```

```
<html>
```

```
<head> ...</head>
```

```
<body>
```

```
<div>
```

Ejemplo de uso del tamaño de fuente en **<strong>em</strong>**

```
</div>
```

```
<strong>Negrita fuera de div</strong>
```

```
</body>
```

```
</html>
```



# UNIDADES DE MEDIDA PARA TEXTOS

## ► Tamaños escalables de fuente

- Utilizar **em** para establecer el tamaño de la fuente.
  - Si el código HTML es el siguiente, ¿cuál es el tamaño de fuente en cada caso?

```
<!DOCTYPE html>
```

```
<html>
```

```
<head> ...</head>
```

```
<body>
```

```
<div>
```

Ejemplo de uso del tamaño de fuente en `<strong>em</strong>`

```
</div>
```

```
<strong>Negrita fuera de div</strong>
```

```
</body>
```

```
</html>
```

$1.5\text{em} * 10\text{pt} = 15\text{pt}$

$2\text{em} * 15\text{pt} = 30\text{pt}$

$2\text{em} * 10\text{pt} = 20\text{pt}$

¿Cuál es el problema?



# UNIDADES DE MEDIDA PARA TEXTOS

---

## ► Tamaños escalables de fuente. **em**

### ► ¿Cuál es el problema?

- **Em** es una unidad acumulativa y por tanto su valor dependerá del valor real de su elemento padre.
- Esto no es un problema especialmente grave para documentos HTML pequeños, pero en documentos HTML grandes y con una gran profundidad en el anidamiento de etiquetas, se hace muy complicado no perder el control sobre los tamaños de los textos y podemos encontrarnos con resultados desagradables.



# UNIDADES DE MEDIDA PARA TEXTOS

---

## ► **Tamaños escalables de fuente (em)**

- Otro ejemplo donde podemos comprobar que puede ser difícil calcular qué medida es exactamente **1em** en un contenedor debido a que dicho tamaño depende de una innumerable cantidad de contenedores que puedan estar anidados.
- Imaginemos la siguiente situación:
  - Tamaño de fuente base: 16px
  - Tamaño deseado para **h1**: 24px.
    - $\text{Target/Context} = 24\text{px} / 16\text{px} = 1.5\text{em}$



# UNIDADES DE MEDIDA PARA TEXTOS

---

- ▶ Ahora dentro de `<h1>` tenemos un `<a>` y queremos un tamaño de letra menor de 12px.
  - ▶ Necesitaríamos saber que h1 son 24px:
    - $1.5em \times 16px = 24px$
    - $12px / 24px = 0.5em$
- ▶ Si dentro del link hay un `<span>` al que queremos darle otro tamaño de fuente distinto, tendríamos que tener en cuenta el tamaño de `<body>` , de `<h1>` y de `<a>` y finalmente calcular el tamaño del `<span>`.
- ▶ Con la unidad de medida % tendríamos los mismos problemas.
- ▶ ¡Con **rem** solo necesitarías saber el tamaño del raíz!
  - ▶ Vamos a verlo...





# UNIDADES DE MEDIDA PARA TEXTOS

## ► Tamaños escalables de fuente. rem

- La mejor unidad para trabajar con textos adaptables es el **rem**.
- Es una versión mejorada de **em** (rem, **root-em**).
- Es un tamaño de fuente relativo al tamaño de fuente que tengamos en el raíz.
- Siguiendo con el ejemplo inicial, si tenemos:

```
Body {font-size: 10pt;}  
div {font-size: 1.5rem;}  
strong {font-size: 2rem;}
```



```
div {  
  font-size: 12px;  
           10pt;  
           0.75em;  
           0.75rem;  
}
```



# UNIDADES DE MEDIDA PARA TEXTOS

## ► Tamaños escalables de fuente

- Así no tenemos un tamaño para la etiqueta `<strong>` que está fuera del `<div>` y otro para la que está dentro del `<div>`.
- La unidad de referencia es el tamaño de la fuente de `<body>`.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head> ...</head>
```

```
<body>
```

```
<div>
```

Ejemplo de uso del tamaño de fuente en `<strong>em</strong>`

```
</div>
```

```
<strong>Negrita fuera de div</strong>
```

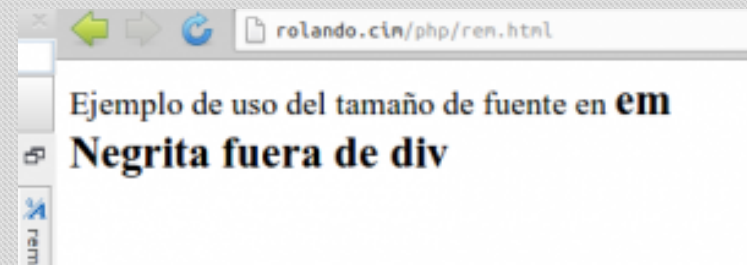
```
</body>
```

```
</html>
```

$1.5\text{rem} * 10\text{pt} = 15\text{pt}$

$2\text{rem} * 10\text{pt} = 20\text{pt}$

$2\text{rem} * 10\text{pt} = 20\text{pt}$



# UNIDADES DE MEDIDA PARA TEXTOS

---

## ▶ Ventajas de rem

- ▶ Nos permite simplificar nuestras **media queries** para adaptarnos a todos los tipos de dispositivos. Bastará con cambiar el **font-size** de la etiqueta `<body>` para que los tamaños de todo el documento cambien en proporción.
- ▶ Rem puede utilizarse en otros atributos como **margin** o **padding** que también dependen del valor del **font-size** de la etiqueta raíz.
- ▶ Todos los navegadores de dispositivos móviles lo soportan.

## ▶ Desventajas de rem

- ▶ No es compatible con navegadores web antiguos.

- ▶ Solución:

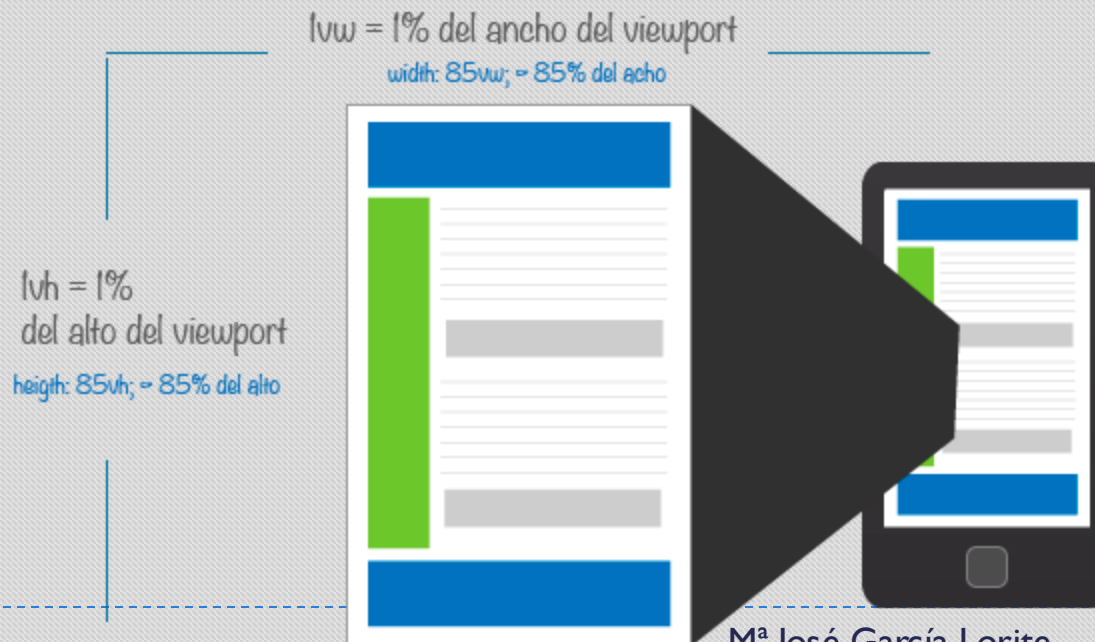
- ☐ `margin: 50px;`
- ☐ `margin: 5rem;`

Si el navegador no soporta rem la 2ª línea la ignora.  
Si el navegador soporta rem la 2ª línea machaca a la primera.  
¡El orden es importante!



# UNIDADES DE MEDIDA PARA TEXTOS

- ▶ viewportwidth (vw) y viewportheight (vh)
  - ▶ Son unidades relativas, aunque en este caso lo son en relación con la pantalla del dispositivo.
  - ▶ **1vw** correspondería a la centésima parte de la anchura del viewport del dispositivo, por lo que **100vw** sería la total.
  - ▶ Algo parecido ocurriría con la vertical **vh** (*solo funciona en móviles*).



# UNIDADES DE MEDIDA PARA TEXTOS

---

## ▶ **viewportwidth (vw) y viewportheight (vh)**

### ▶ **VENTAJA**

- ▶ Utilizar % para las alturas es un quebradero de cabeza. Para tener las alturas controladas hay que darle dimensiones a todos los contenedores hasta llegar al elemento donde queremos definir su altura con un valor porcentual. Por este motivo, la unidad **vh** es bastante útil.

### ▶ **INCONVENIENTE**

- ▶ **vw** y **vh** NO corresponden exactamente con las dimensiones reales de la pantalla del dispositivo, puesto que en ocasiones el viewport puede falsear los datos y el dispositivo puede tener unas dimensiones distintas de pantalla.
- ▶ Para controlar el viewport tenemos:

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

[http://www.w3schools.com/css/css\\_rwd\\_viewport.asp](http://www.w3schools.com/css/css_rwd_viewport.asp)



# LAS CAPAS. ATRIBUTOS

---

## ▶ **Atributo visibility**

- ▶ Sirve para especificar si la capa está oculta o visible al usuario.
- ▶ Puede tomar los siguientes valores:

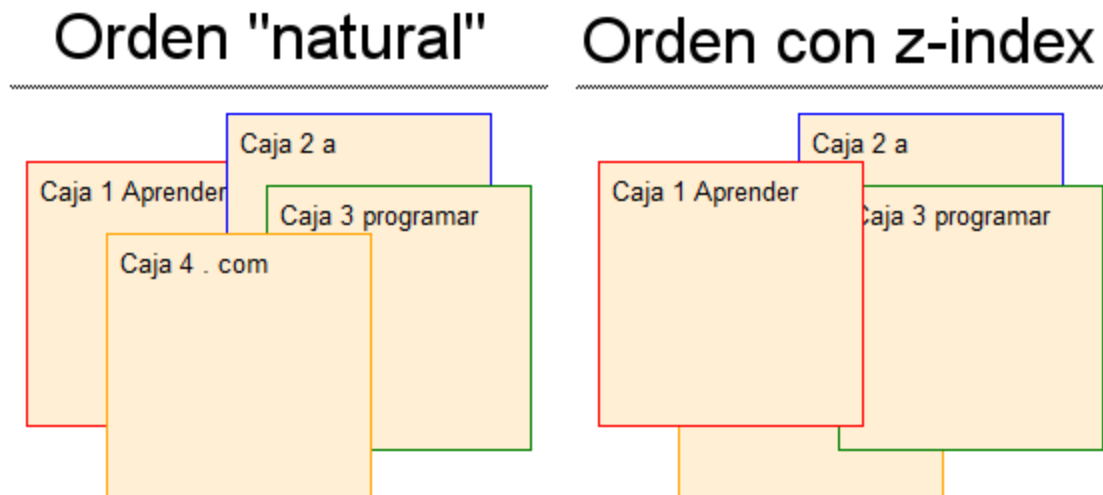


- ☐ **Visible**
- ☐ **Hidden** (oculta)
- ☐ **Inherit** (heredada) Es el valor por defecto.
  - ☐ Quiere decir que la capa hereda la visibilidad de la capa donde se encuentra metida.
  - ☐ Si la capa no está dentro de ninguna otra capa, se supone que se encuentra dentro de la capa documento <body> que por defecto está visible.
- ☐ **Collapse**
  - ☐ Solamente para elementos de tabla. Sirve para ocultar una fila o una columna de una tabla. Si se utiliza en cualquier otro elemento funciona como Hidden.
- ☐ **Initial**
  - ☐ Para coger el valor inicial.

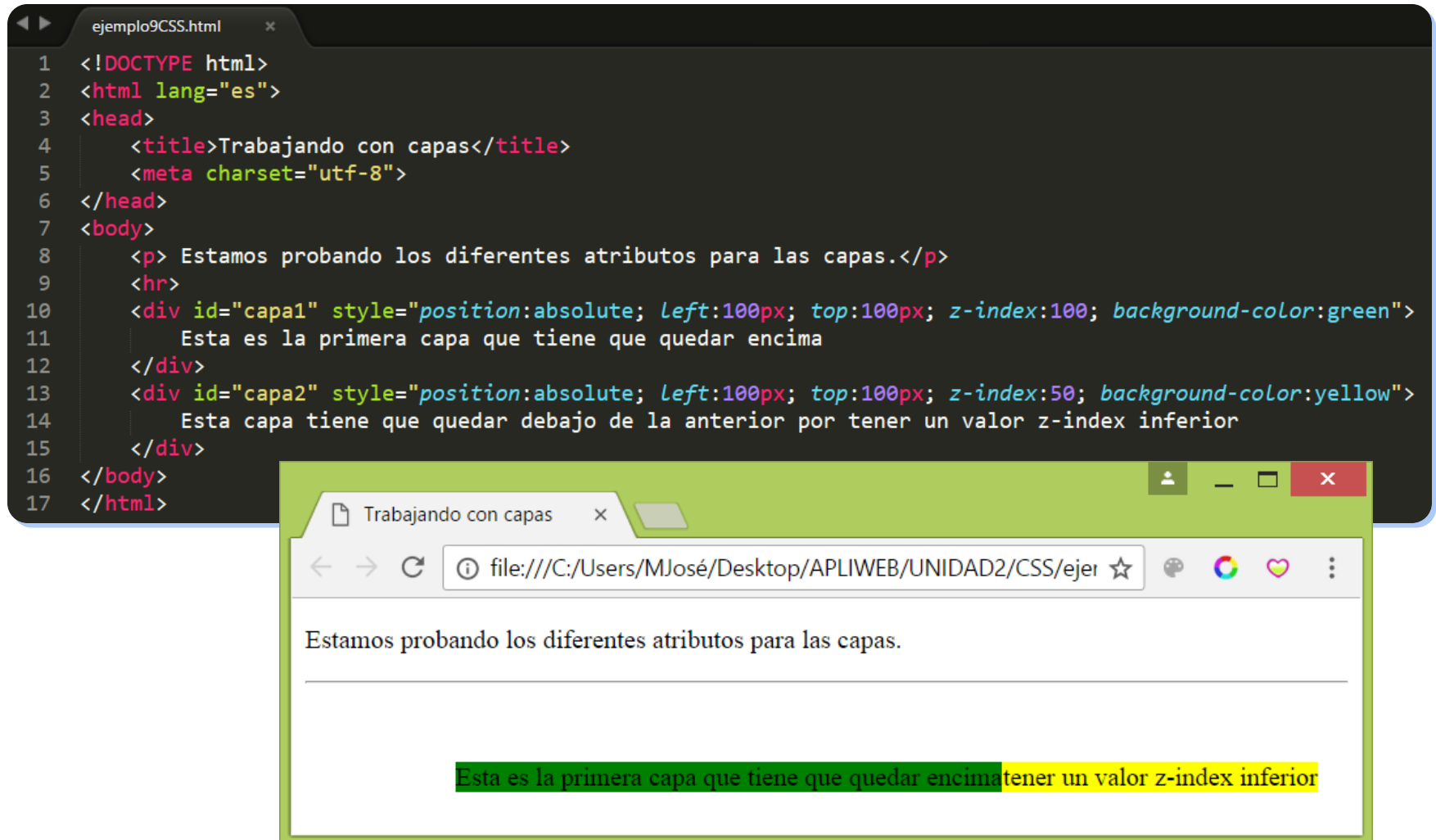
# LAS CAPAS. ATRIBUTOS

## ► Atributo z-index

- Sirve para indicar la posición sobre el ejeZ que tendrán las distintas capas de la página.
- En el caso de capas superpuestas, podemos especificar qué capas se verán encima o debajo de otras.
- Este atributo toma **valores enteros**.
- **Cuanto mayor sea el valor, más arriba se verá la capa.**



# LAS CAPAS. ATRIBUTOS



The image displays a code editor window titled 'ejemplo9CSS.html' and a web browser window showing the rendered HTML. The code defines two absolute layers: 'capa1' (green, z-index: 100) and 'capa2' (yellow, z-index: 50). The browser shows the text 'Estamos probando los diferentes atributos para las capas.' followed by a horizontal line, and then the text 'Esta es la primera capa que tiene que quedar encima' on a green background and 'tener un valor z-index inferior' on a yellow background.

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <title>Trabajando con capas</title>
5   <meta charset="utf-8">
6 </head>
7 <body>
8   <p> Estamos probando los diferentes atributos para las capas.</p>
9   <hr>
10  <div id="capa1" style="position:absolute; left:100px; top:100px; z-index:100; background-color:green">
11    Esta es la primera capa que tiene que quedar encima
12  </div>
13  <div id="capa2" style="position:absolute; left:100px; top:100px; z-index:50; background-color:yellow">
14    Esta capa tiene que quedar debajo de la anterior por tener un valor z-index inferior
15  </div>
16 </body>
17 </html>
```

Trabajando con capas

file:///C:/Users/MJosé/Desktop/APLIWEB/UNIDAD2/CSS/ejer

Estamos probando los diferentes atributos para las capas.

Esta es la primera capa que tiene que quedar encima tener un valor z-index inferior

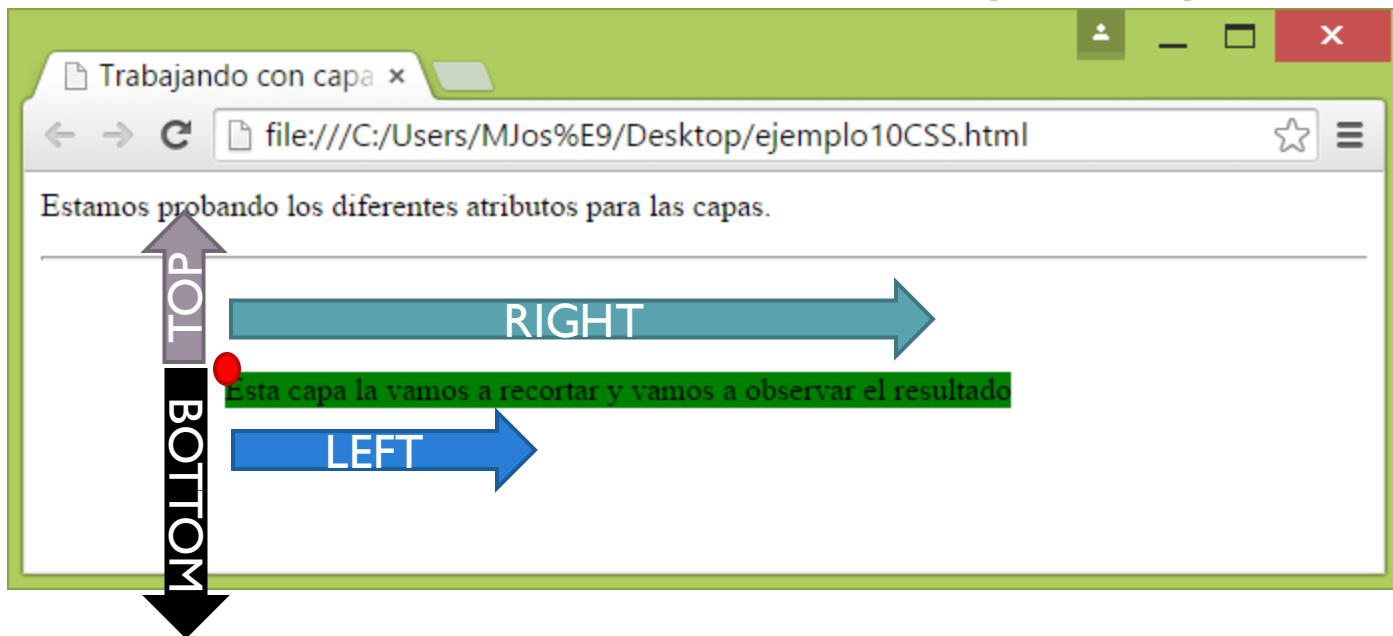


# LAS CAPAS. ATRIBUTOS

**DEPRECATED**

## ► Atributo clip

- Sirve para recortar determinadas áreas de una capa; así podremos ocultar una parte de una capa que sea visible.
- El **clipping** se indica por medio de 4 valores, enteros con esta sintaxis:
  - ❑ **clip: rect (top, right, bottom, left)**
  - ❑ Los valores enteros indican la distancia desde el **punto top-left** de la capa

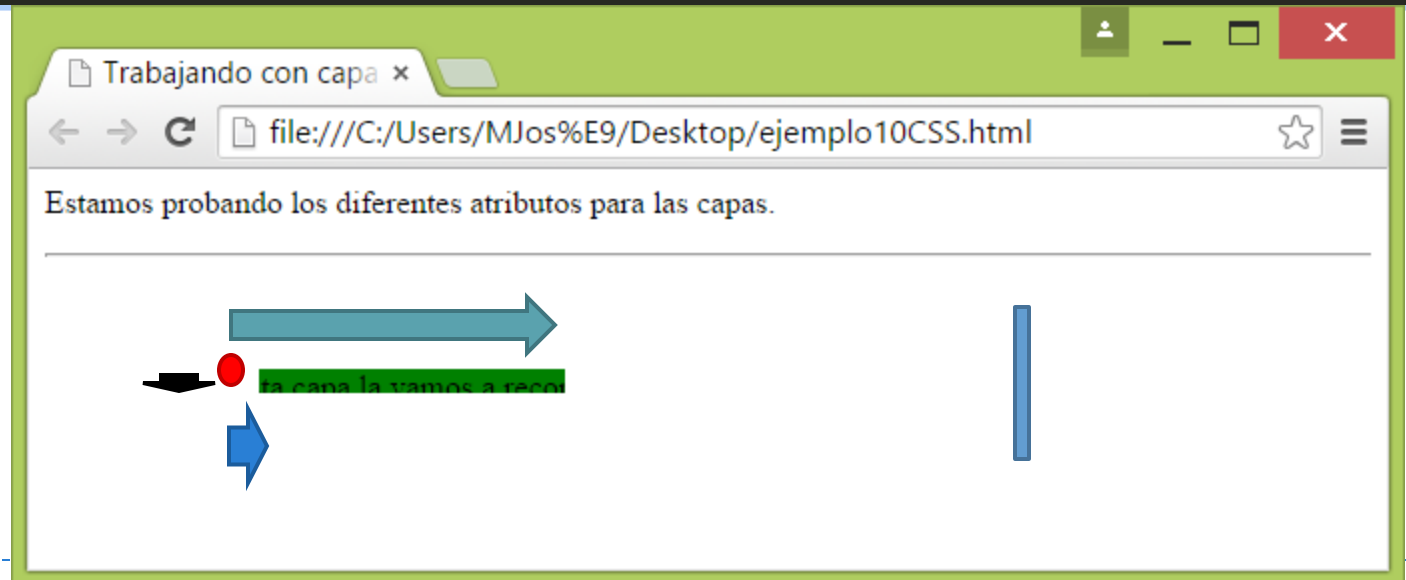


# LAS CAPAS. ATRIBUTOS

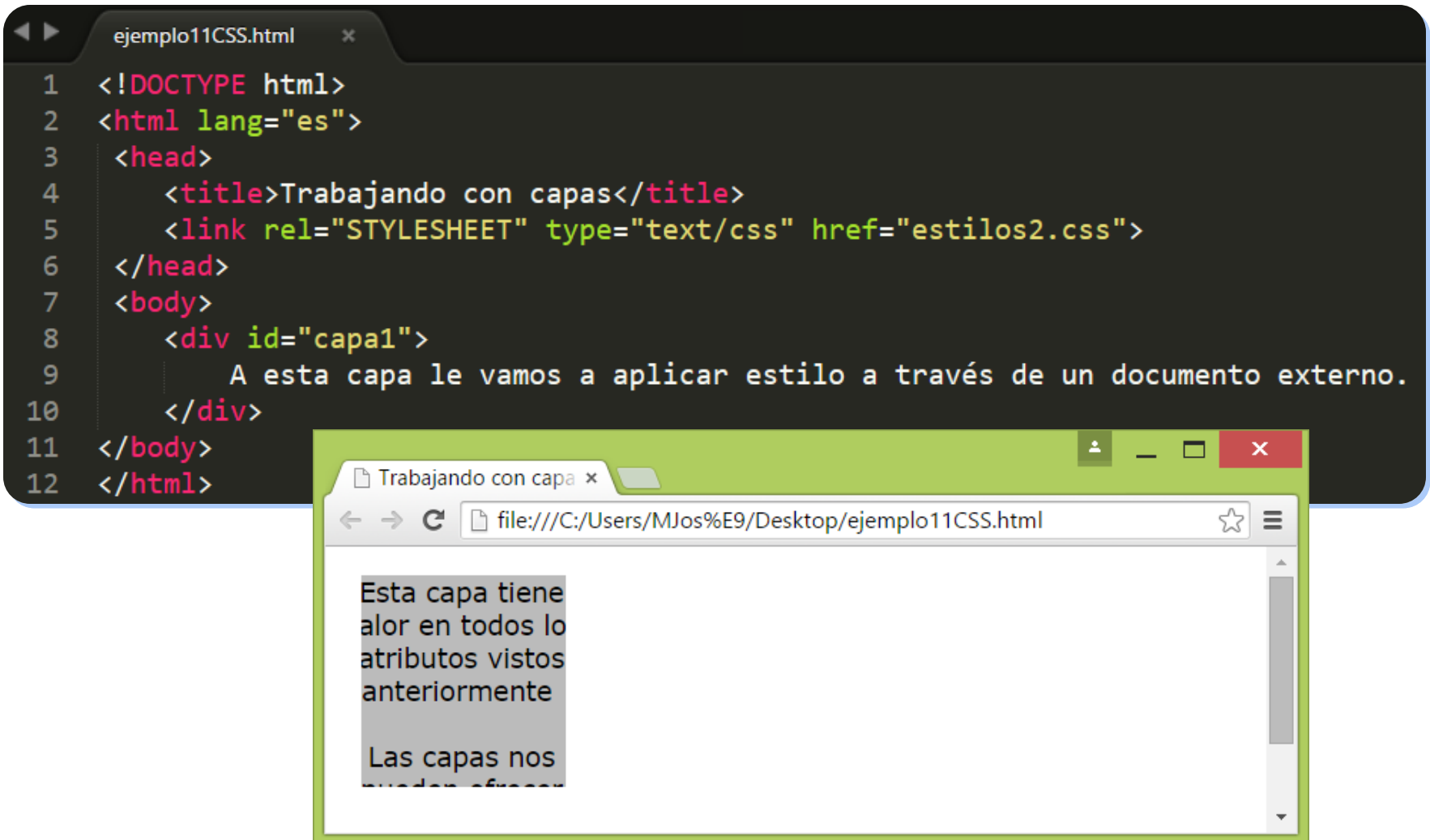
ejemplo10CSS.html

```
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Trabajando con capas</title>
5     <meta charset="utf-8">
6   </head>
7   <body>
8     <p>Estamos probando los diferentes atributos para las capas.</p>
9     <hr>
10    <div id="capa1" style="position:absolute; left:100px; top:100px; background-color:green; clip: rect(0,168,12,15)">
11      Esta capa la vamos a recortar y vamos a observar el resultado
12    </div>
13  </body>
14 </html>
```

top, right, bottom, left



# LAS CAPAS. ATRIBUTOS



# LAS CAPAS. ESTILOS EXTERNOS

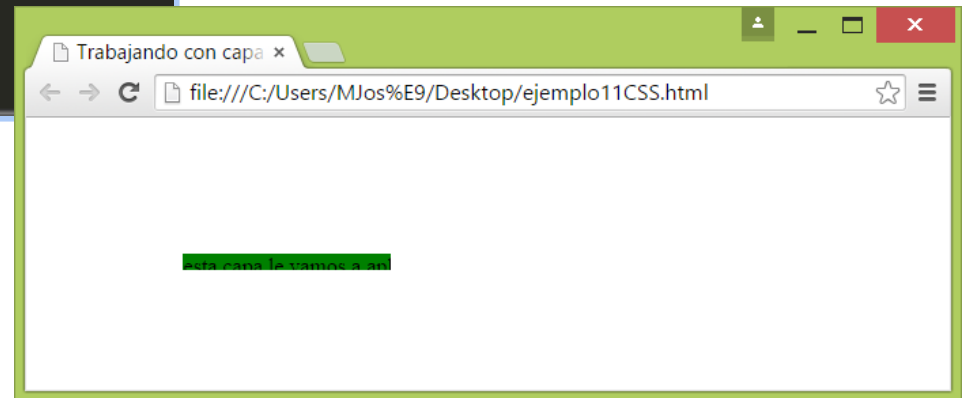
- Podemos asignar estilo a las capas a través de un documento externo.

```
ejemplo11CSS.html x
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Trabajando con capas</title>
5     <link rel="stylesheet" type="text/css" href="estilos2.css">
6   </head>
7   <body>
8     <div id="capa1">
9       A esta capa le vamos a aplicar estilo a través de un documento externo.
10    </div>
11  </body>
12 </html>
```

# LAS CAPAS. ESTILOS EXTERNOS

- ▶ En el fichero de estilos, se escribe el carácter almohadilla #, seguido del **identificador** indicado en la etiqueta y entre llaves { } los atributos CSS que deseemos.

```
estilos2.css
1  #capa1{
2      position:absolute;
3      left:100px;
4      top:100px;
5      background-color:green;
6      clip: rect(0,168,12,15)
7  }
```



# ESTILOS Shorthand

---

- ▶ Sirven para escribir de forma reducida reglas CSS para que los archivos CSS sean menos pesados y más fáciles de actualizar.

- ▶ **Propiedad font**

- ▶ font-style || font-variant || font-weight || font-size / line-height || family-font

- ▶ Ejemplo:

- **p**{font: italic normal bold 12px/14pt Verdana,Tahoma,Arial}

- ▶ **Propiedad background**

- ▶ background-color || background-image || background-repeat || background-attachment || background-position

- ▶ Ejemplo:

- **body**{background: #FFF url(..images/ejemplo.gif) no-repeat fixed center}

# ESTILOS Shorthand

## ► Propiedad margin

► longitud | porcentaje | auto

► Ejemplos:

□ **body {margin: 5px}**

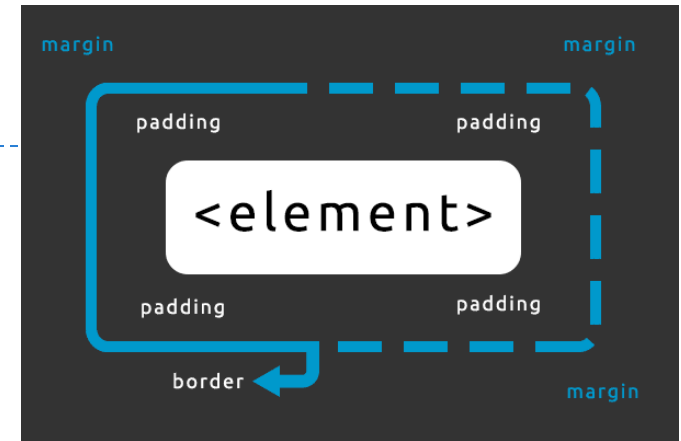
□ Todos los márgenes a 5px

□ **p{margin: 2px 4px}**

□ Márgenes superior e inferior a 2px, márgenes izquierdo y derecho a 4px

□ **div{margin: 1px 2px 3px 4px}**

□ Margen superior a 1px, margen derecho a 2px, margen inferior a 3px y margen izquierdo a 4px



# ESTILOS Shorthand

## ► Propiedad padding

► longitud | porcentaje | auto

► Ejemplos:

❑ `body{padding: 2em 3em 4em 5em}`

❑ Si definimos cuatro valores estamos aplicando el padding superior, derecho, inferior e izquierdo

❑ `body{padding: 2em 4em}`

❑ Si definimos dos o tres valores, los valores faltantes se toman del lado opuesto: superior e inferior a 2em, derecho e izquierdo a 4em

❑ `body{padding: 5em}`

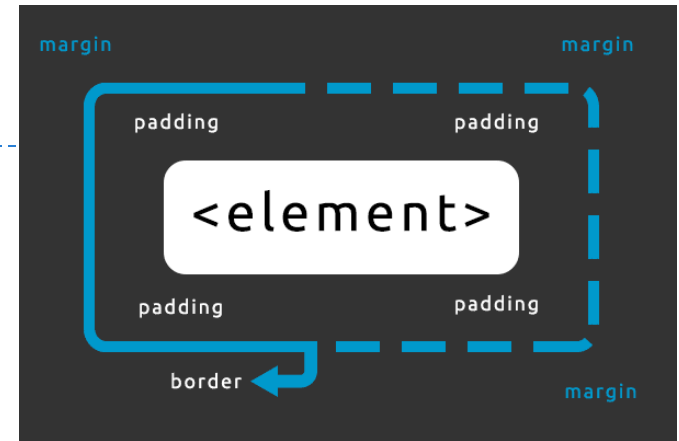
❑ Si definimos un solo valor se aplican a todos los lados.

## ► Propiedad Border

► `border-width || border-style || color`

► Ejemplo:

❑ `h3{border: thick dotted blue}`

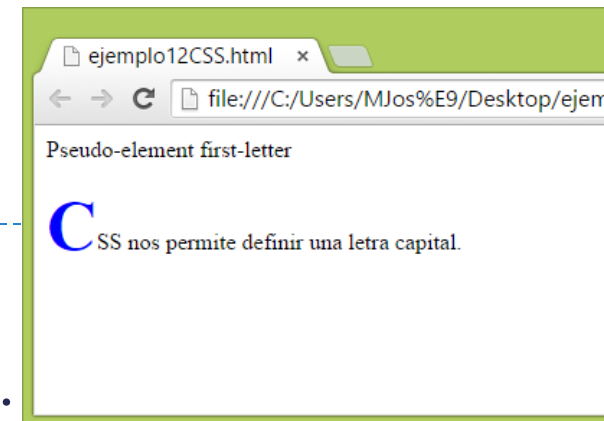




# Pseudo-elementos en CSS

## ► Pseudo-elemento first-letter

- Permite poner una letra capital a un párrafo.
- Ejemplo de utilización:



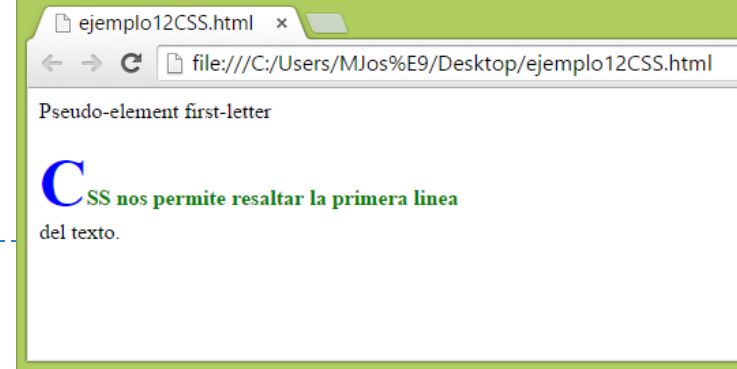
```
ejemplo12CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Pseudo-element first-letter</title>
5     <meta charset="utf-8">
6     <link rel="stylesheet" type="text/css" href="estilos3.css">
7   </head>
8   <body>
9     <p>CSS nos permite resaltar la primera linea del texto.</p>
10  </body>
11 </html>
```

```
ejemplo12CSS.html  estilos3.css
4 p:first-letter{
5   font-size: 300%;
6   color:blue;
7   font-weight: bold;
8 }
```

# Pseudo-elemento en CSS

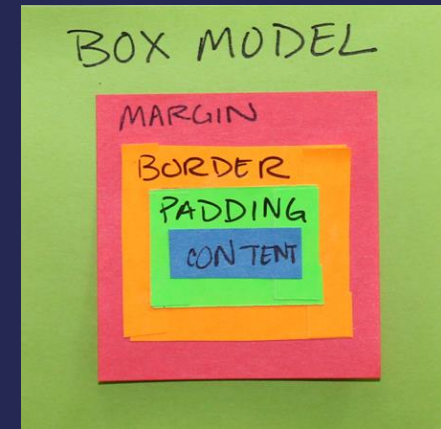
## ► Pseudo-elemento first-line

- Podemos aplicar un estilo propio a la primera línea del texto.
- Ejemplo de utilización:



```
ejemplo12CSS.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Pseudo-element first-letter</title>
5     <meta charset="utf-8">
6     <link rel="stylesheet" type="text/css" href="estilos3.css">
7   </head>
8   <body>
9     <p>CSS nos permite resaltar la primera línea del texto.</p>
10  </body>
11 </html>

ejemplo12CSS.html  estilos3.css
4 p:first-letter{
5   font-size: 300%;
6   color:blue;
7   font-weight: bold;
8 }
9 p:first-line{
10  font-size: 100%;
11  color:green;
12  font-weight: bold;
13 }
```



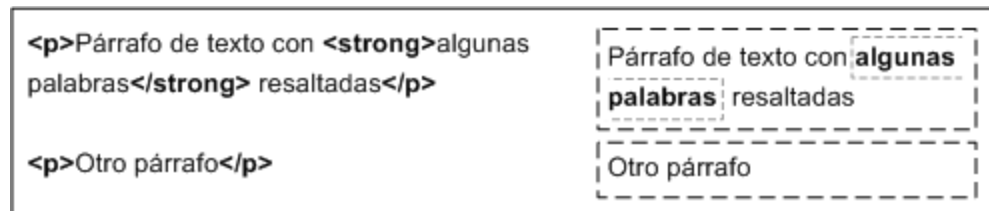
# MODELO DE CAJAS DE CSS

BOX MODEL

# CONCEPTO

---

- ▶ El modelo de cajas es la característica **más importante** de CSS ya que condiciona el diseño de todas las páginas web.
- ▶ El modelo de cajas es el comportamiento de CSS que hace que **todos los elementos de las páginas web se representen mediante cajas rectangulares**.
- ▶ Cada vez que se inserta una etiqueta HTML, se crea automáticamente una nueva caja rectangular (invisible) que encierra los contenidos de ese elemento.



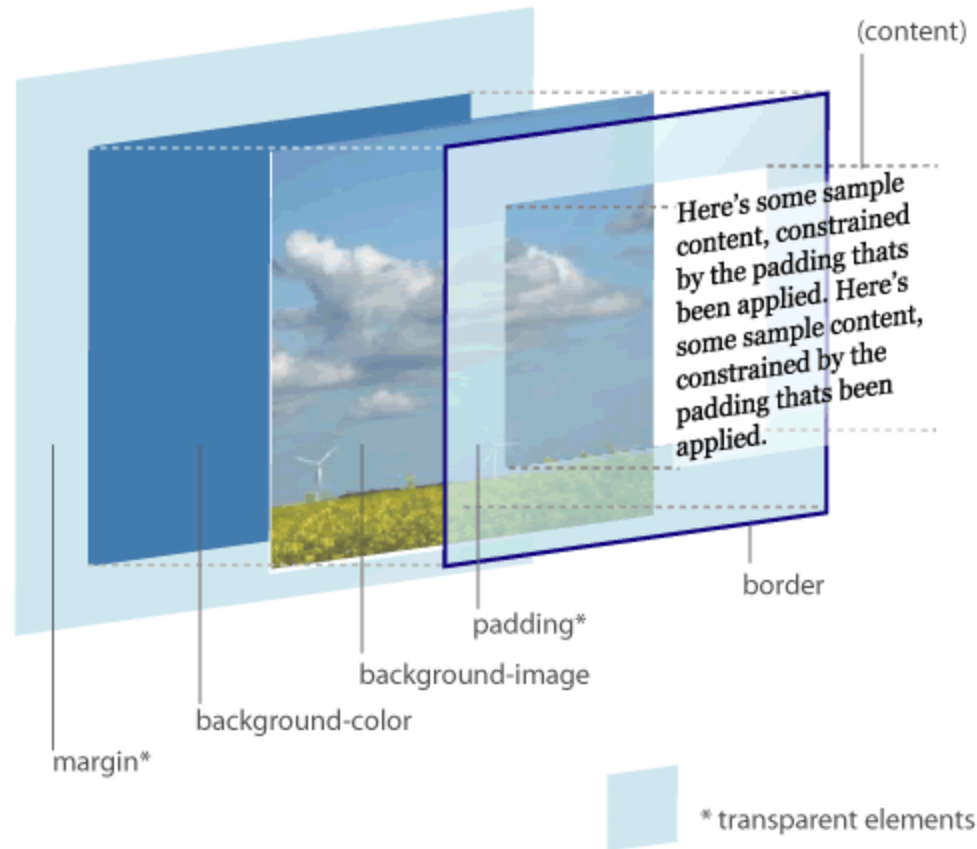
# CONCEPTO

---

- ▶ 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 del elemento (texto, imágenes etc.)
  - ▶ **Relleno** (*padding*): espacio libre opcional existente entre el contenido y el borde.
  - ▶ **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 del espacio de relleno.
  - ▶ **Color de fondo** (*background color*): color que se muestra por detrás del contenido y del espacio de relleno.
  - ▶ **Margen** (*margin*): separación opcional existente entre la caja y el resto de cajas adyacentes.

# CONCEPTO

## THE CSS BOX MODEL HIERARCHY



- Los navegadores crean y colocan las cajas de forma automática, pero CSS permite modificar todas sus características.

# CONCEPTO

---

- ▶ El relleno (padding) y el margen son transparentes, por lo que dejan ver el color y/o imagen de fondo.
- ▶ Si una caja define tanto un color como una imagen de fondo, la imagen tiene más prioridad y es la que se visualizará.
- ▶ No obstante, si la imagen no cubre totalmente la caja o si la imagen tiene zonas transparentes, también se visualiza el color de fondo.
- ▶ *Combinando imágenes transparentes y colores de fondo se pueden lograr efectos gráficos muy interesantes.*

# CONCEPTO

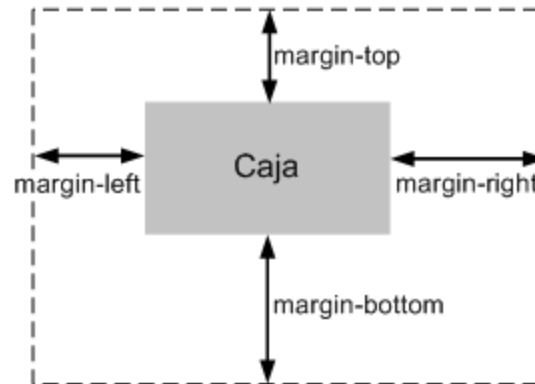
- ▶ Vamos a ver detenidamente los atributos margin, border, padding y content.





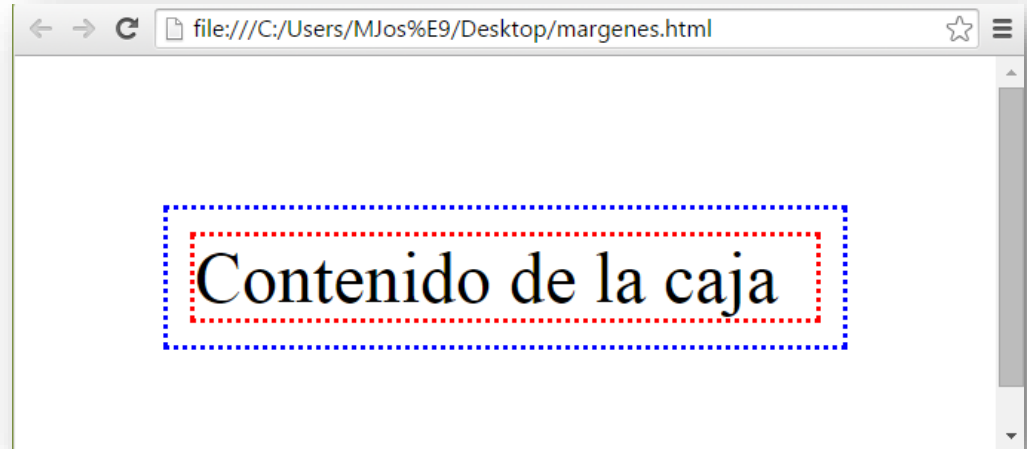
# ATRIBUTOS MARGIN

- ▶ CSS define 4 atributos para los márgenes:



- ▶ Cada uno de estos atributos establece la separación entre el borde lateral de la caja y el resto de cajas adyacentes.
- ▶ Las unidades más utilizadas para indicar los márgenes de un elemento son los píxeles (cuando se requiere una precisión total), los em y los rem (para hacer diseños que mantengan las proporciones) y los porcentajes (para hacer diseños fluidos).

```
margenes.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Modelo de Caja</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       body {
8         margin-top: 100px;
9         margin-right: 100px;
10        margin-bottom: 100px;
11        margin-left: 100px;
12        border: 3px dotted blue;
13      }
14      div{
15        margin-top: 15px;
16        margin-right: 15px;
17        margin-bottom: 15px;
18        margin-left: 15px;
19        border: 3px dotted red;
20        font-size:0.5in;
21      }
22    </style>
23  </head>
24  <body>
25    <div> Contenido de la caja </div>
26  </body>
27 </html>
```



Al no haber relleno (padding) la distancia entre el borde azul y el rojo es de 15px.

La distancia entre el borde azul y los bordes de la vista del navegador es de 100px.

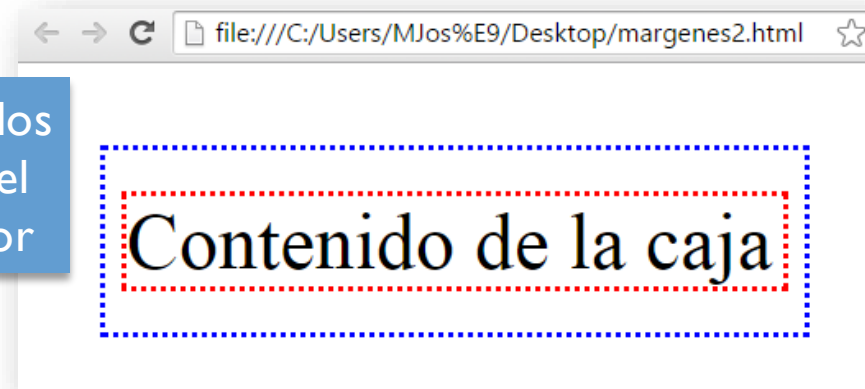
Si cambiáis los valores de los márgenes podréis entender mejor el concepto de margen.

# ATRIBUTOS MARGIN

- ▶ El código anterior lo podemos simplificar utilizando el atributo **margin**:

```
margenes2.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Modelo de cajas</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       body {
8         margin: 50px;
9         border: 3px dotted blue;
10      }
11      div{
12        margin: 25px 10px 25px 10px;
13        border: 3px dotted red;
14        font-size:0.5in;
15      }
16    </style>
17  </head>
18  <body>
19    <div> Contenido de la caja </div>
20  </body>
21 </html>
```

Para todos los márgenes el mismo valor



El 1<sup>er</sup> valor se corresponde con el margen superior y sigue el sentido de las agujas del reloj.



# ATRIBUTOS MARGIN

---

- ▶ El atributo **margin** admite entre uno y cuatro valores, con el siguiente significado:
  - ▶ Si solo se indica un valor, todos los márgenes tienen ese valor.
  - ▶ Si se indican dos valores, el primero se asigna al margen superior e inferior y el segundo se asigna a los márgenes izquierdo y derecho.
  - ▶ Si se indican tres valores, el primero se asigna al margen superior, el tercero se asigna al margen inferior y el segundo valor se asigna los márgenes izquierdo y derecho.
  - ▶ Si se indican los cuatro valores, el orden de asignación es: margen superior, margen derecho, margen inferior y margen izquierdo.

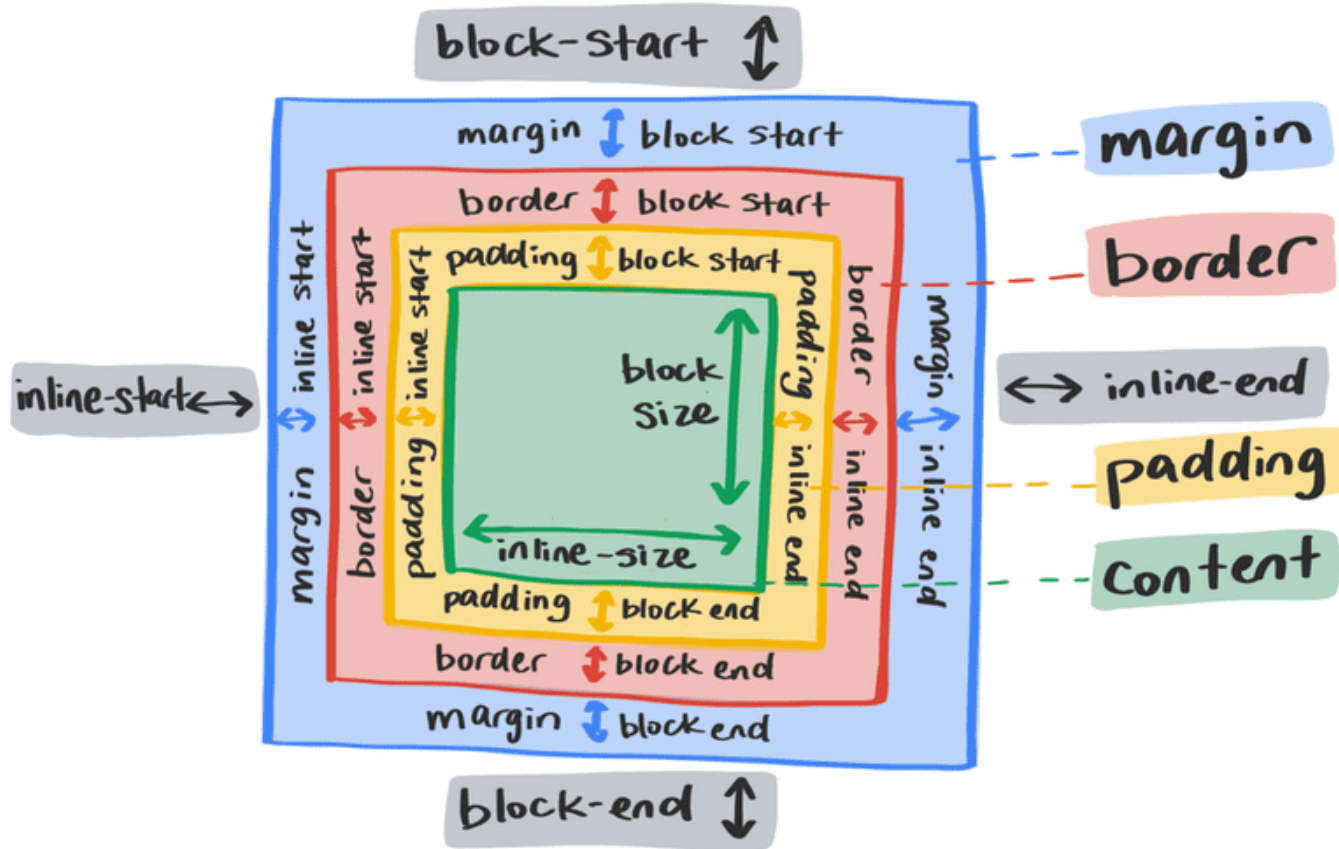


# ATRIBUTOS MARGIN

---

- ▶ Los valores de estos atributos (y de cualquier de los siguientes que veremos), también pueden ser auto o inherit.
- ▶ **Auto** calcula automáticamente la mínima distancia según la relación con otros elementos.
  - ▶ Esto es útil cuando las relaciones se hacen con medidas relativas.
- ▶ **Inherit** hereda el valor del mismo atributo en la caja que lo contiene, es decir, hereda los valores del padre.

# NUEVOS ATRIBUTOS MARGIN



# NUEVOS ATRIBUTOS MARGIN

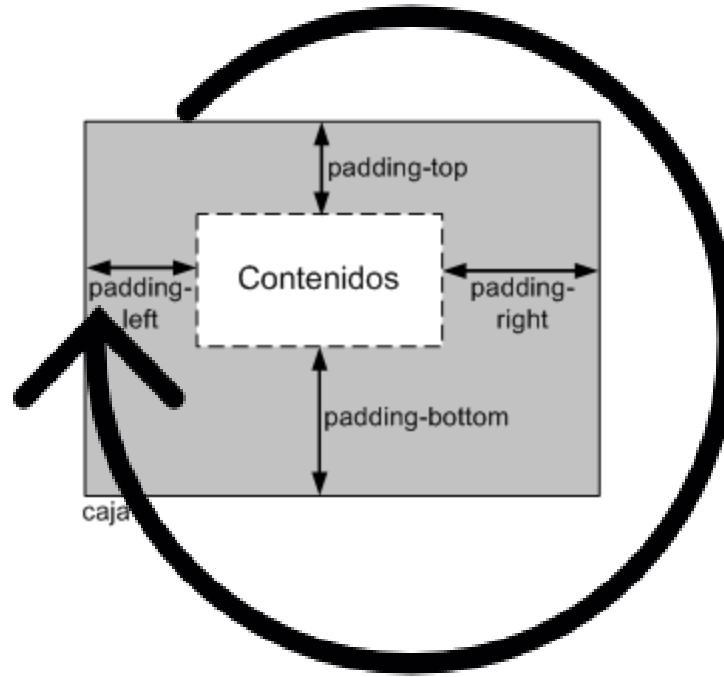
---

- ▶ La diferencia entre **margin-block-start** y **margin-top** está en cómo manejan los márgenes en función de la dirección en la que se escribe el texto, lo que afecta a su **compatibilidad internacional**.
  - **margin-top**: específico para esquemas de escritura horizontales (de izquierda a derecha o de derecha a izquierda). No es flexible si el esquema de escritura cambia. Siempre aplica el margen en la parte superior, independientemente de la dirección del texto.
  - **margin-block-start**: se adapta al esquema de escritura. Controla el margen al **inicio del bloque** en función del esquema de escritura del texto:
    - En un esquema de escritura **horizontal** sería igual que **margin-top**.
    - En un esquema de escritura **vertical** (como el japonés, que puede ir de arriba a abajo), este margen correspondería al "inicio" del bloque que sería el lado izquierdo.

# ATRIBUTOS PADDING

---

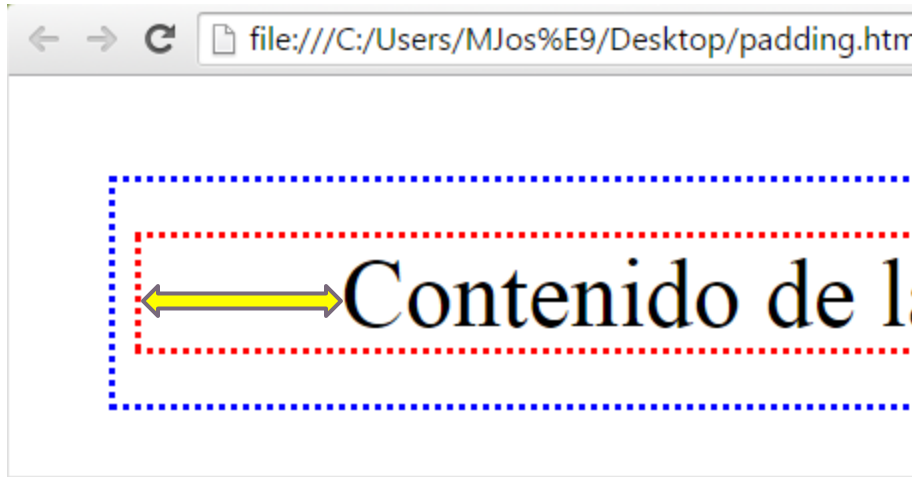
- ▶ El **Padding** o relleno es la distancia entre el borde y el contenido.



- ▶ Los valores de este atributo son los mismos que para el atributo **margin**.



# ATRIBUTOS PADDING



```
padding.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <title>Modelo de cajas</title>
5     <meta charset="utf-8">
6     <style type="text/css">
7       body {
8         margin: 50px;
9         border: 3px dotted blue;
10      }
11      div{
12        margin: 25px 10px 25px 10px;
13        border: 3px dotted red;
14        padding-left: 100px;
15        font-size:0.5in;
16      }
17    </style>
18  </head>
19  <body>
20    <div> Contenido de la caja </div>
21  </body>
22 </html>
```

# ATRIBUTOS PADDING

---

- ▶ Al igual que el atributo margin, el atributo **padding** simplifica estos 4 atributos:
  - ▶ padding: 10px;
    - ▶ Esto creará un "hueco interno" de 10 píxeles alrededor de la capa.
  - ▶ padding: 1px 2px 3px 4px;
    - ▶ Esto creará un hueco de 1 píxel por arriba, de 2 píxeles por la derecha, de 3 píxeles por abajo y de 4 píxeles por la izquierda.
  - ▶ padding: 10px 5px;
    - ▶ Esto creará un hueco de 10 píxeles por arriba y por abajo y de 5 píxeles por los lados.

# ATRIBUTOS BORDER

---

- ▶ Definen el estilo y el color del borde de la caja.
  - ▶ `border: 3px dotted red;`
    - ▶ Borde rojo punteado de 3 píxeles.
  - ▶ Es posible separar la asignación de color y estilo:
    - ▶ `border-color`
    - ▶ `border-style`
  - ▶ También se puede especificar un color y estilo para cada uno de los bordes **por separado**:
    - ▶ `border-top`
    - ▶ `border-bottom`
    - ▶ `border-right`
    - ▶ `border-left`

<http://librosweb.es/referencia/css/border.html>

# ATRIBUTOS BORDER

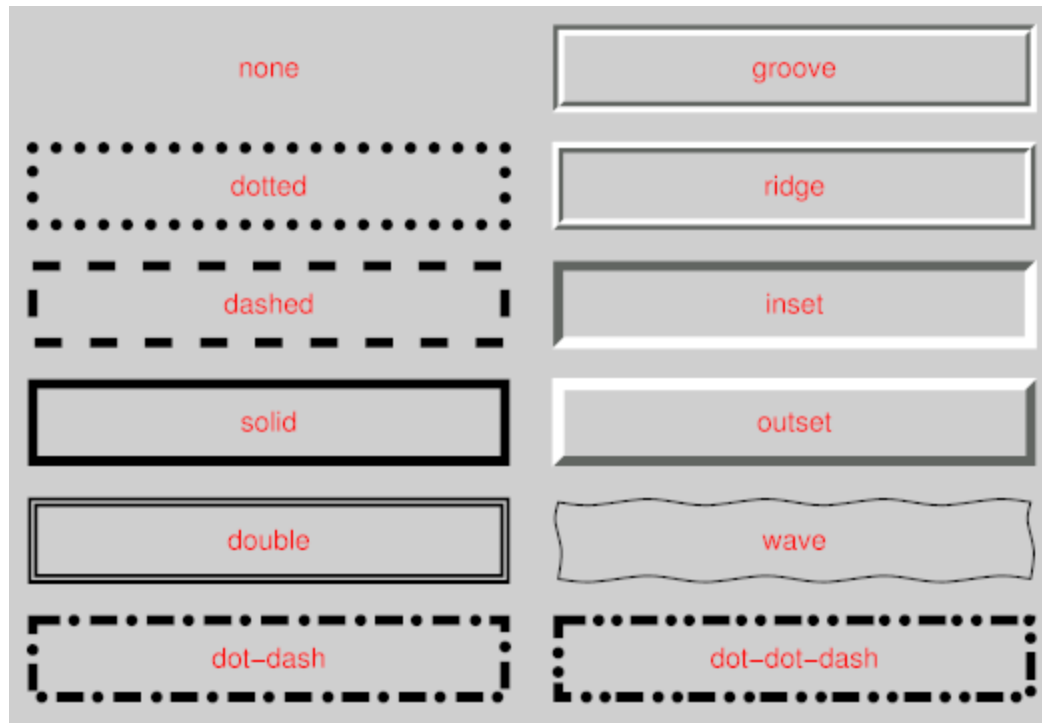
---

- ▶ Es posible separar la asignación de color y estilo:
  - ▶ `border-color`
  - ▶ `border-style`
  - ▶ `border-width`
- ▶ También se puede especificar un color y estilo para cada uno de los bordes **por separado**:
  - ▶ `border-top: 1px solid #C00`
    - `border-top-width: 1px;`
    - `border-top-style: solid;`
    - `border-top-color: #C00;`
  - ▶ `border-bottom`
  - ▶ `border-right`
  - ▶ `border-left`

# ATRIBUTOS BORDER

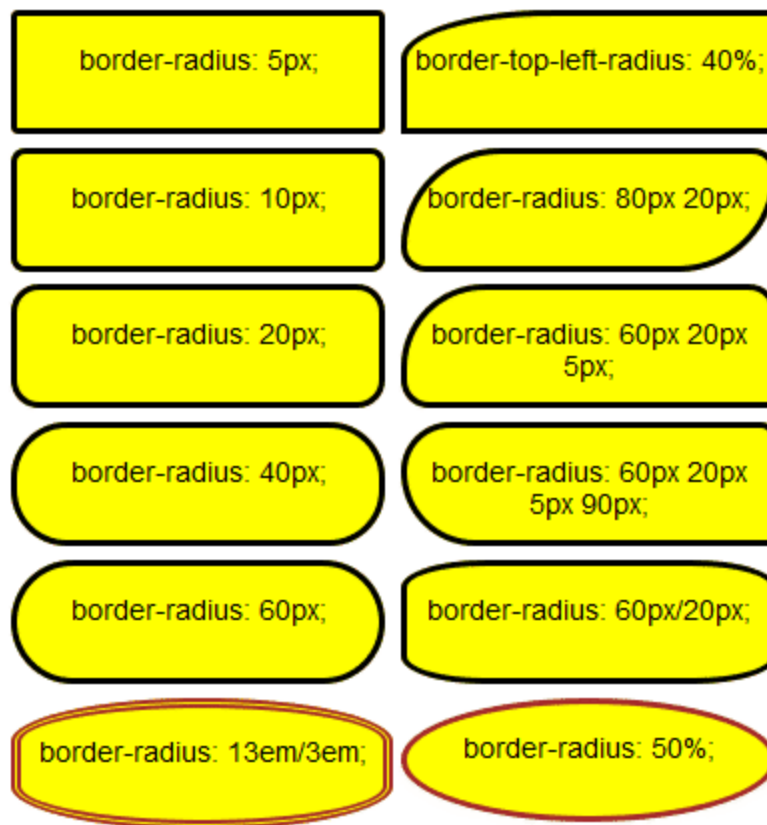
---

## ► Estilo del borde:



# ATRIBUTOS BORDER

- ▶ El atributo **border-radius** nos permite hacer bordes redondeados:



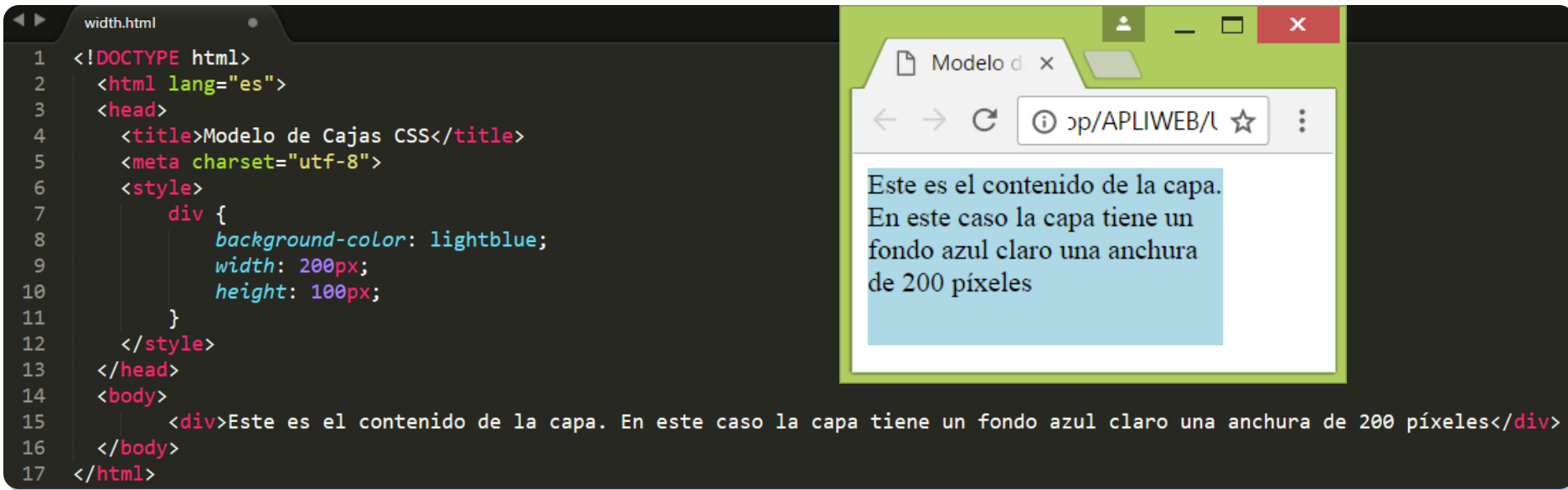
```
border.html
1  <!DOCTYPE html>
2  <html lang="es">
3    <head>
4      <title>Atributos border</title>
5      <meta charset="utf-8">
6      <style type="text/css">
7        body{
8          margin: 100px;
9          border-style: inset;
10         border-color: blue;
11         border-radius: 15px;
12         border-width: thick;
13         padding: 15px;
14       }
15       div{
16         margin-top: 15px;
17         margin-right: 25px;
18         margin-bottom: 15px;
19         margin-left: 25px;
20         padding-left: 10px;
21         border-top: 3px dotted red;
22         border-right: 2px solid blue;
23         border-bottom: 3px double green;
24         border-left: 3px groove red;
25       }
26     </style>
27   </head>
28   <body>
29     <div> Texto contenido dentro de la capa </div>
30   </body>
31 </html>
```

Texto contenido dentro de la capa

# Atributos del contenido

## ► Atributos **width** y **height**

- Para establecer la anchura y la altura respectivamente del contenido de la caja.

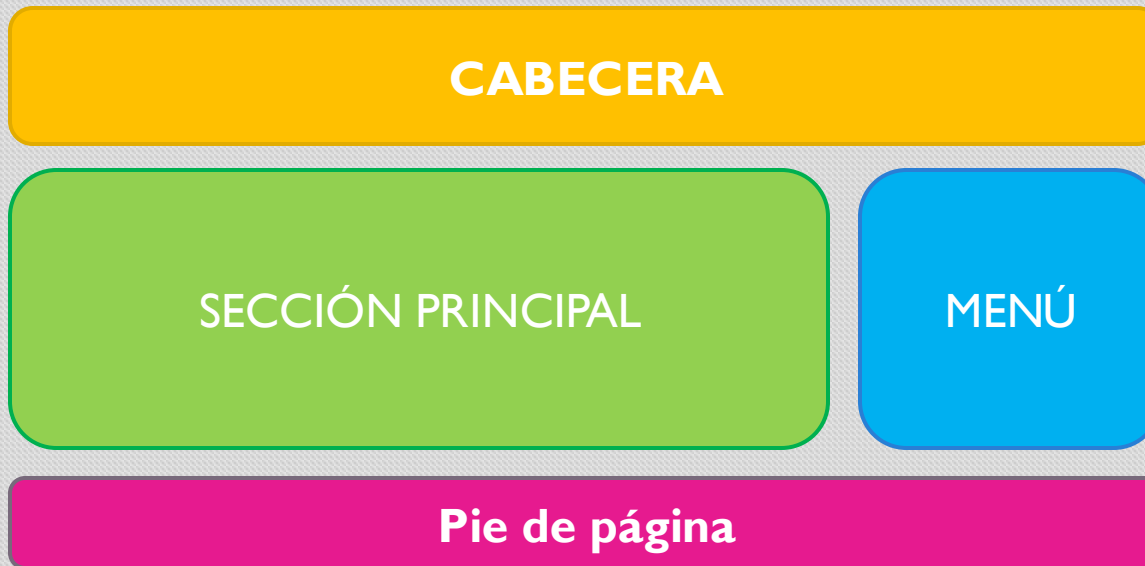




# EJERCICIO 4

---

- ▶ Vamos a realizar la siguiente estructura básica de página web haciendo uso de capas:



```

36 <body>
37   <div id="cabecera">
38     <header><h1>Esta es la cabecera</h1></header>
39   </div>
40
41   <div id="cuerpo">
42     <section>
43       <h2>Sección principal</h2>
44       <p>Esta es la sección principal de nuestra página</p>
45     </section>
46   </div>
47
48   <div id="barra">
49     <aside>
50       <nav>
51         <ul>
52           <li><a href="uno.html">Item de menú 1</a> </li>
53           <li><a href="dos.html">Item de menú 2</a> </li>
54           <li><a href="tres.html">Item de menú 3</a> </li>
55         </ul>
56       </nav>
57     </aside>
58   </div>
59
60   <div class='pie'>
61     <footer>Pie de página</footer>
62   </div>
63 </body>
64 </html>

```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Utilizando el modelo de capas</title>
5   <meta charset="utf-8">
6   <style>
7     body {
8       padding: 10px;
9     }
10    #cabecera {
11      width: 100%;
12      height: 50px;
13      background-color: #FFBF00;
14    }
15    #cuerpo {
16      width: 70%;
17      height: 200px;
18      float: left;
19      background-color: #58FA82;
20    }
21    #barra {
22      width: 30%;
23      height: 200px;
24      float: left;
25      background-color: #58D3F7;
26    }
27    .pie {
28      width: 100%;
29      height: 50px;
30      clear: both;
31      background-color: #FA58D0;
32    }
33  </style>
34 </head>
```

Indica que la caja se coloque a la izquierda

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



# OTROS ATRIBUTOS

ATRIBUTOS DE FUENTES, PÁRRAFOS, FONDOS, TABLAS, ...

# Atributos de fuentes

---

- ▶ **color**

- ▶ RGB o nombre de color.

- ▶ **font-size**

- ▶ unidades, xx-small, x-small, small, medium, large, x-large, xx-large.

- ▶ **font-family**

- ▶ serif, sans-serif, cursive, fantasy, monospace.

- ▶ **font-weight**

- ▶ normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900.

- ▶ **font-style**

- ▶ normal, italic, oblique.

# Atributos de párrafos

---

## ▶ **line-height**

- ▶ Define el interlineado.
- ▶ Valores: normal, unidades

## ▶ **text-decoration**

- ▶ Valores: none, underline, overline, line-through

## ▶ **text-align**

- ▶ Valores: left, right, center, justify.

## ▶ **text-indent**

- ▶ Para aplicar una sangría.
- ▶ Valores: unidades.

## ▶ **text-transform**

- ▶ Valores: capitalize, uppercase, lowercase, none text-transform, none.

# Atributos de fondo

---

- ▶ **background-color**

- ▶ Para indicar el color de fondo de la página.
- ▶ Valor: RB o nombre del color.

- ▶ **background-image**

- ▶ Para indicar la imagen de fondo de la página.
- ▶ Valor: ruta de la imagen (absoluta o relativa).



# Atributos tablas

---

- ▶ **caption-side**

- ▶ Posición del título.
- ▶ Valores: top o bottom.

- ▶ **table-layout**

- ▶ Valores: auto o filled.

- ▶ **border-collapse**

- ▶ Valores: collapse o separate.

- ▶ **border-spacing**

- ▶ Espaciado entre celdas adyacentes.
- ▶ Valor: unidades

- ▶ **empty-cells**

- ▶ Visibilidad de los bordes de las celdas sin contenido.
- ▶ Valores: show o hide

# RECURSO CSS3

---

- ▶ Hay muchísimos atributos CSS además de los enumerados anteriormente.
- ▶ Todos los podrás encontrar en el siguiente recurso que nos va a resultar de gran utilidad durante el desarrollo de esta unidad:



<http://www.w3schools.com/cssref/>