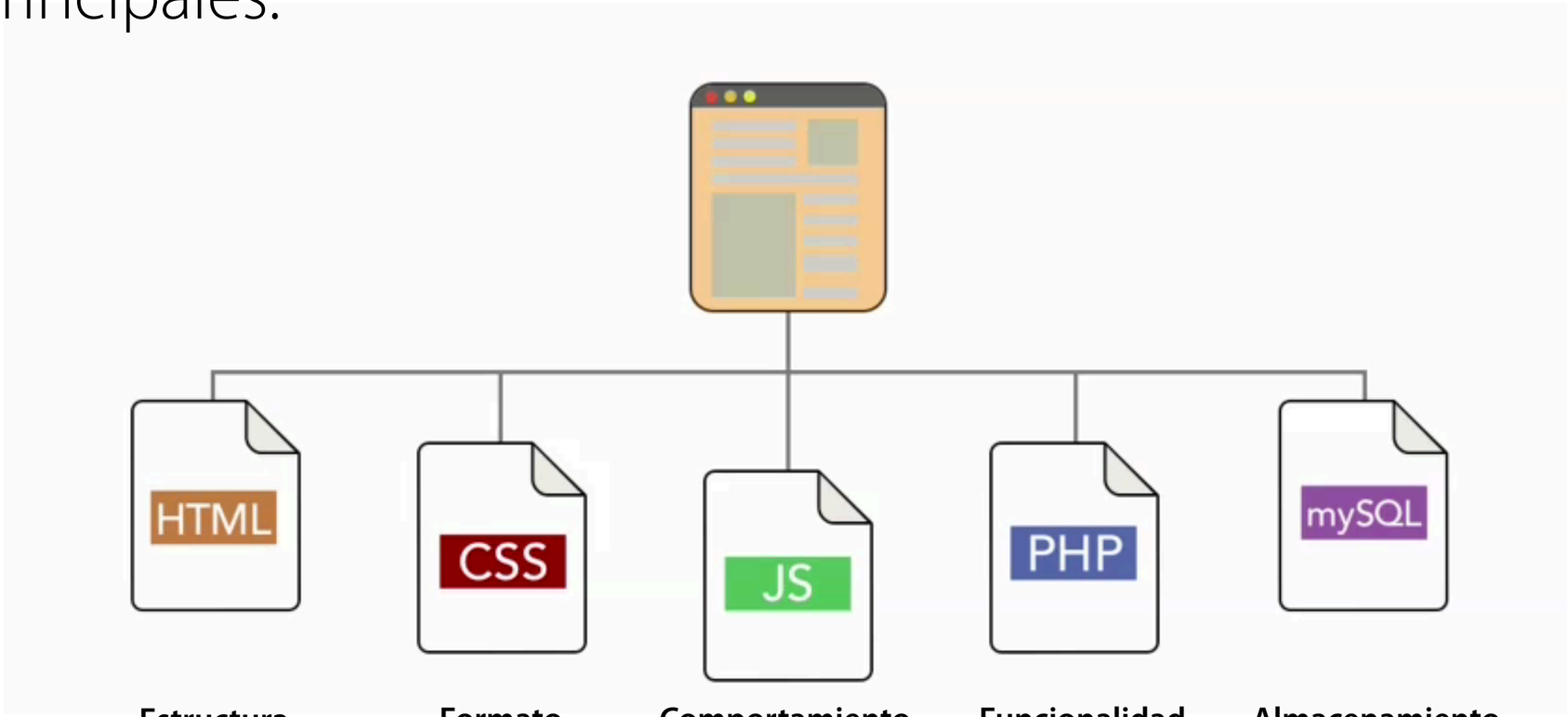
Fundamentos CSS

¿Qué es el CSS?

- Las siglas CSS corresponden a *Cascade Style Sheet*:
  Hojas de estilo en cascada
- El HTML aporta estructura a la página.
- El **CSS** es un conjunto de reglas de formato encargado de controlar la presentación del código HTML.

Componentes de una Web

- Una página web puede estar formada por 5 componentes principales:



Componentes de una Web

- Para crear una página web estática sólo necesitaremos HTML para la estructura y CSS para la presentación:

Características del CSS

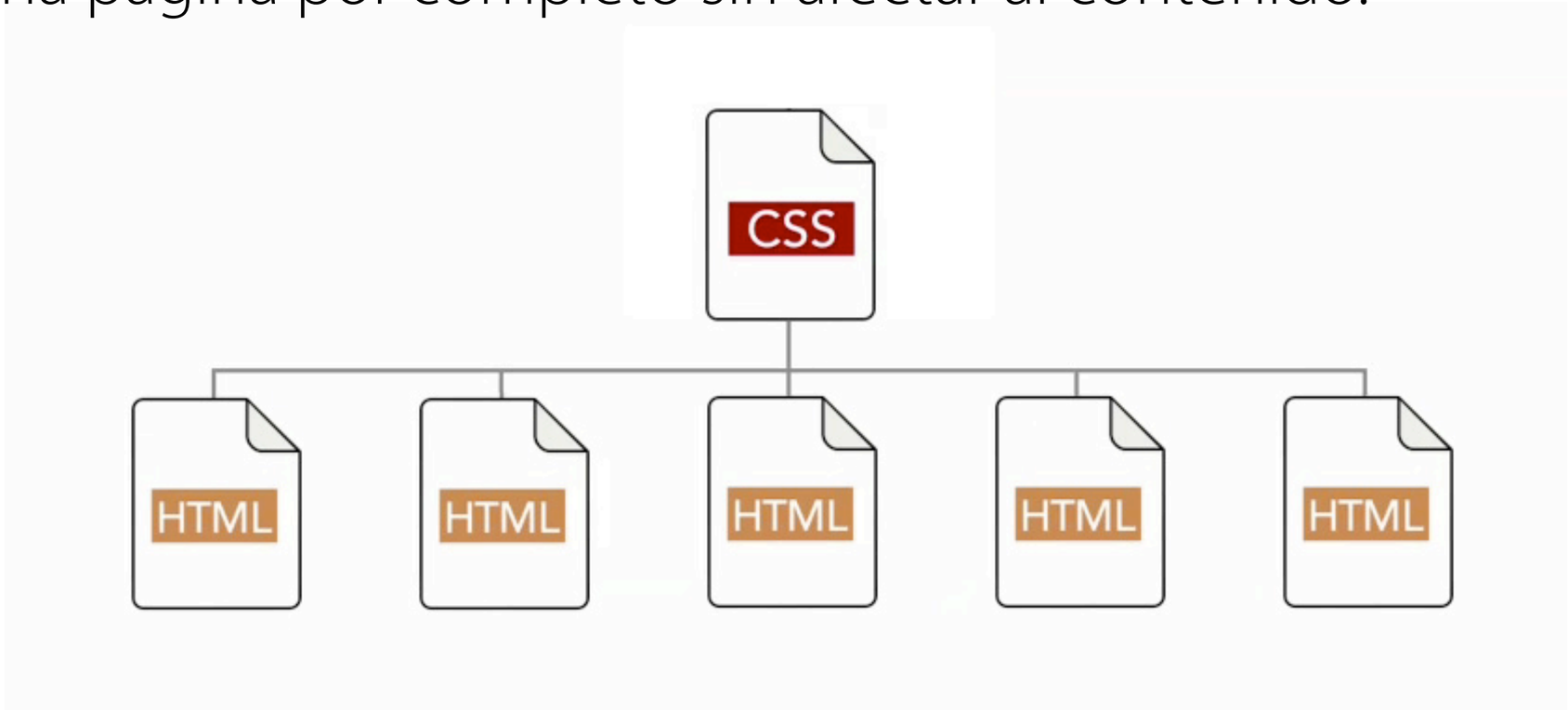
- El CSS es un **conjunto de reglas de formato** más que un lenguaje de programación como el HTML o Javascript.
- Generalmente se encuentran en **archivos externos** y se vinculan al encabezado de la web <head>.
- El término "**en cascada**" hace referencia a cómo se aplican los estilos: Según el orden, un estilo prevalece sobre el otro.

Estilos “en cascada”

- Los estilos se aplican en función del orden en el que se encuentran en la página.
- Prevalecen los estilos aplicados en función de su especificidad y el orden en el que se aplican.
- Aplicar un estilo significa **seleccionar** un elemento de la página y modificar sus atributos por defecto.

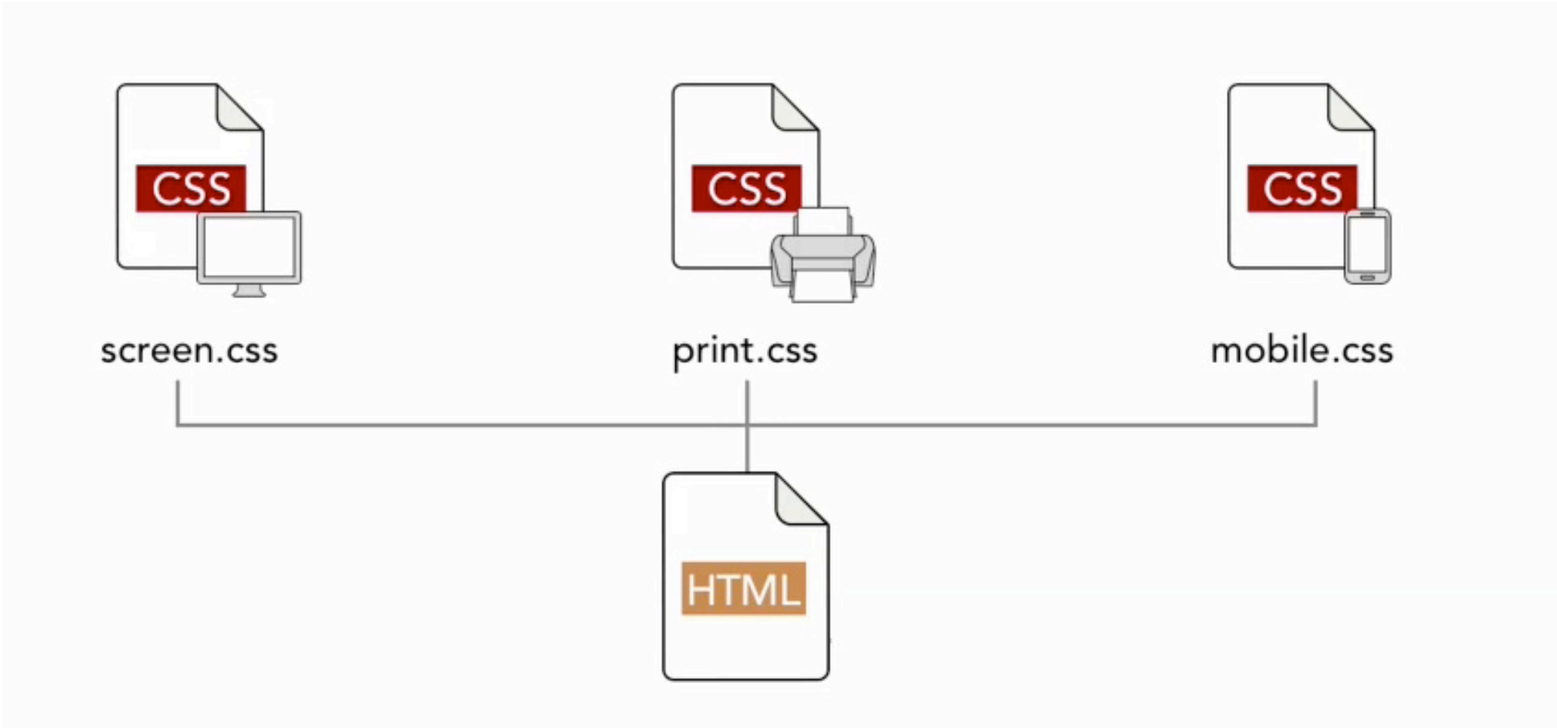
Beneficios del CSS

- El carácter modular del CSS permite modificar el aspecto de una página por completo sin afectar al contenido.



Beneficios del CSS

- Mediante hojas de estilo específicas podemos optimizar contenidos para múltiples dispositivos.



Estilos CSS por defecto

Estilos del navegador web

- Un documento HTML sin hoja de estilos adjunta se presenta con una hoja de estilos por defecto aplicada por **el navegador web**.
- Cada navegador web dispone de su propia hoja de estilos que pueden variar entre sí.
- Una buena práctica para evitar inconsistencias es aplicar un **reset** a esa hoja de estilos.

Reset CSS

```
    html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p,
blockquote, pre, a, abbr, acronym, address, big,
cite, code, del, dfn, em, font, img,
ins, kbd, q, s, samp, small, strike,
strong, sub, sup, tt, var, dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
center, u, b, i {
     margin: 0;
     padding: 0;
     border: 0;
     outline: 0;
     font-weight: normal;
     font-style: normal;
     font-size: 100%;
     font-family: inherit;
     vertical-align: baseline
}

body {
     line-height: 1
}
```

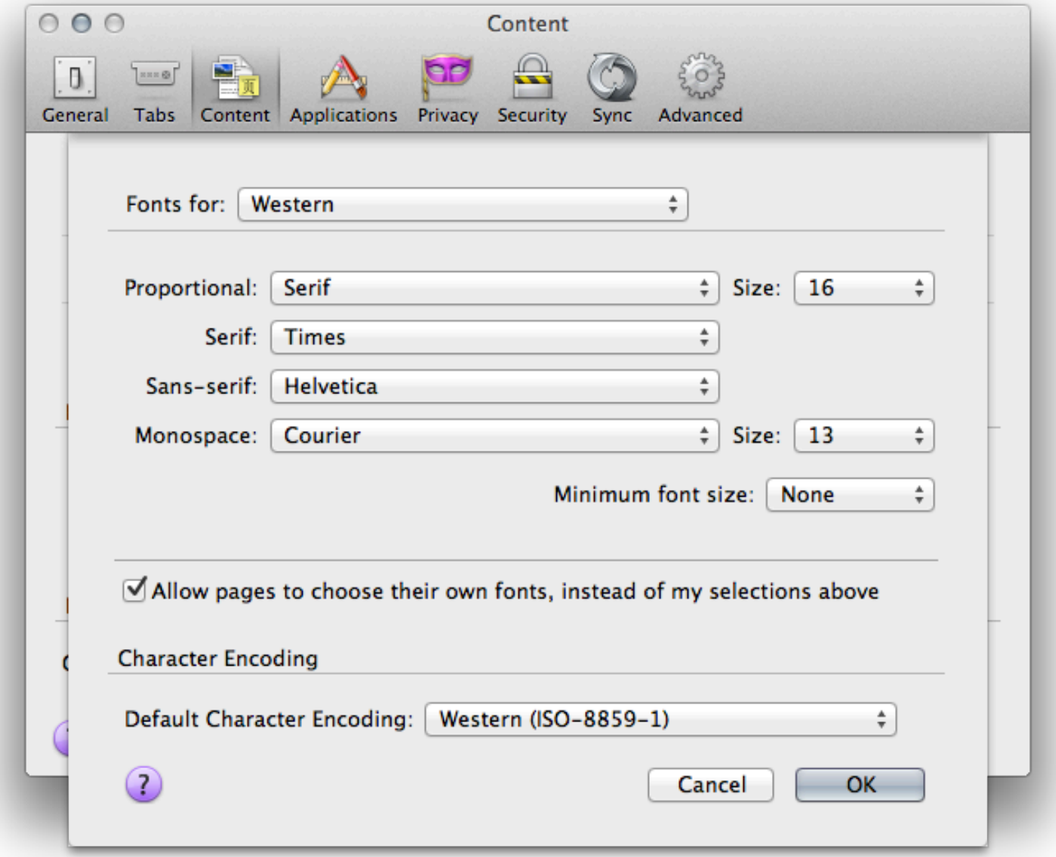
Vía Anieto2k:
http://www.anieto2k.com/2007/08/07/reseteando-estilos-css/

Reset CSS

- Mediante el **Reset CSS** nos aseguramos de que todos los elementos de la página se representen con el mismo valor en todos los navegadores.
- El Reset CSS sobreescribe los estilos por defecto del navegador web.
- Posteriormente, la **hoja de estilos del documento** sobreescribirá los valores del Reset CSS.
  (Los estilos se aplican en cascada).

Estilos de usuario

- Todos los navegadores web permiten al usuario establecer unos **estilos por defecto**.
- Los estilos de usuario definen los **valores por defecto**.
- El usuario puede omitir por completo la hoja de estilos del documento.



Firefox 12 MacOSX

Sintaxis CSS

Sintaxis CSS

- **Selector**
- Declaración
- Propiedad
- Valor

```
p {
    font-size: 1em;
    color: #000;
    }
```

Mediante el **Selector** indicamos al código el elemento al que queremos aplicar formato.

Sintaxis CSS

- Selector
- **Declaración**
- Propiedad
- Valor

```
p {
    font-size: 1em;
    color: #000;
    }
```

La **Declaración** (entre {}) es la secuencia de estilos que aplicaremos al **Selector**.

Sintaxis CSS

- Selector
- Declaración
- **Propiedad**
- Valor

```
p {
    font-size: 1em;
    color: #000;
}
```

La **Propiedad** es la característica o atributo del **Selector** que queremos modificar. Finaliza siempre en **:**

Sintaxis CSS

- Selector
- Declaración
- Propiedad
- **Valor**

```
p {
    font-size: 1em;
    color: #000;
}
```

Modificando el **Valor** de una propiedad estamos aplicando un estilo personalizado al selector en cuestión. Finaliza siempre en **;**

Sintaxis CSS

Espacios en blanco

```
p  {
     font-size: 1em;
     color: #000;
     }
```

=

```
p  {font-size: 1em; color: #000;}
```

Las dos formulaciones son correctas. En este caso los espacios en blanco y saltos de línea se utilizan para hacer más legible el código.

Sintaxis CSS

Espacios en blanco

```
p  #autor  {
    font-size: 1em;
    color: #000;
    }
```

≠

```
p#autor {
    font-size: 1em;
    color: #000;
    }
```

En este caso, el espacio en blanco **modifica el tipo de selector**.
El selector de la primera declaración es menos específico que en la segunda.

Sintaxis CSS

Comentarios de código CSS

```
/* Inicio de estilos del Menú principal */
#menu  {
      font-size: 1em;
      }
```

En la primera línea que precede a la declaración de #menu indicamos que pertenece al menú principal mediante un comentario que nos permitirá revisar el código más fácilmente.

Para añadir un comentario ha de tener el formato:

/* ...comentario... */

Sintaxis CSS

Comentarios de código CSS

```
p  #autor  {
      /* font-size: 1em; */
      color: #000;
      }
```

Otra utilidad de los comentarios es la de anular temporalmente la aplicación de un estilo concreto en una línea de la declaración.

En el ejemplo anulamos el primer atributo **font-size: 1em;** mientras **color: #000;** no se ve afectado.

Sintaxis CSS

Comentarios de código CSS

```
/*
p  #autor  {
      font-size: 1em;
      color: #000;
      }
*/
```

Todas las declaraciones contenidas entre el signo de apertura de comentario /* y el de cierre */ quedarán anuladas y no tendrán efecto en la hoja de estilos.

Selectores CSS

Selectores CSS

Mediante los selectores podemos **aplicar estilos CSS** de forma genérica a todos los elementos de una página o sólo a uno muy específico.

Tipos de Selectores CSS

Selector de Etiqueta (o elemento)

```
p {
    font-size: 1em;
    color: #000;
    }
```

```
h1 {
    font-family: Helvetica, Arial, san-serif;
    color: #000;
    }
```

El selector de etiqueta es el **más genérico**. Indicamos la etiqueta que queremos aplicar el estilo y lo hará a todos los elementos de la página que se correspondan con esa etiqueta.

Tipos de Selectores CSS

Selector Clase

HTML

```
<h1 class=”titulo”>
    Hola mundo
</h1>
```

CSS

```
.titulo {
    font-family: Helvetica, Arial, san-serif;
    color: #000;
    }
```

· Las clases son atributos HTML que se pueden asignar a cualquier elemento.
· Podemos asignar el nombre que deseemos (sin espacios ni acentos).
· Se pueden aplicar a múltiples elementos en una misma página.
· Son sensibles a las mayúsculas.

Tipos de Selectores CSS

Selector ID

HTML

```
<div id=”sidebar”>
    ... contenido ...
</div>
```

CSS

```
#sidebar {
    margin: 10px;
    }
```

- Los ID son atributos HTML que se pueden asignar a cualquier elemento.
- Podemos utilizar el nombre que deseemos (sin espacios ni acentos).
- Los ID difieren de las clases en que han de ser **únicos** en una misma página.

Combinaciones de Selectores CSS

Combinando selectores podemos **aumentar la especificidad** para seleccionar únicamente un tipo concreto de elemento bajo una determinada circunstancia.

Combinaciones de Selectores CSS

Selector Elemento-Específico

```
div#sidebar {
    margin: 10px;
}
```

```
h2.subtitulo {
    font-weight: bold;
}
```

La combinación **Elemento-específico** aplicará una clase o un ID exclusivamente al elemento que le precede.
Hay que procurar **no** dejar espacios en blanco en el selector.

Combinaciones de Selectores CSS

Selector Descendente

```
div h2 span  {
    font-weight: bold;
    }
```

· Los **selectores descendientes** permiten especificar un elemento basándose en su ubicación dentro de otro elemento.
· Este tipo de selector ofrece altos niveles de especificidad pero es poco versátil.
· Se colocan las etiquetas una a continuación de la otra separadas por espacio en blanco.

Combinaciones de Selectores CSS

Agrupación de selectores

```
h1 { font-weight: bold; }
h2 { font-weight: bold; }
.titulo { font-weight: bold; }
```

=

```
h1, h2, .titulo  {
    font-weight: bold;
    }
```

Podemos agrupar selectores que requieran el mismo formato situando uno a continuación del otro separados por coma (**,**).

De esta forma minimizamos la cantidad de código y creamos una hoja de estilos más eficiente.

Selectores contextuales

Selectores contextuales

Selector universal

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

El selector universal * (asterisco) aplica la declaración a todos los elementos de la página sin excepción.

Selectores contextuales

Selector de atributo

```
input[type] {
    font-size: 14px;
    }
```

Selector atributo

```
input[type="search"] {
    font-size: 14px;
    }
```

Selector atributo=”valor”

El selector de atributo aplicará la declaración sólo cuando el elemento disponga de un atributo que coincida exactamente con el contenido entre paréntesis [ ] incluya o no valor.

Selectores contextuales

Selector de hijos

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

Selector con espacios

=

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

Selector sin espacios

El selector contextual de hijos seleccionan a un elemento cuando se da la relación especificada. En el ejemplo anterior quedarán seleccionados los elementos "**p**" que son hijos del contenedor "**div**".

El espacio en blanco en el selector no afecta al resultado.

Selectores contextuales

Selector de hermanos adyacentes

```
h1 + h2 {
    margin-top: -0.5em;
    }
```

El selector contextual de hermanos adyacentes seleccionará a un elemento cuando se da la relación especificada. En el ejemplo anterior quedarán seleccionados los elementos "**h2**" cuando se encuentren inmediatamente después de un "**h1**".

Pseudo-clases y pseudo-elementos

Los selectores vistos hasta ahora actúan sobre elementos que podemos encontrar en un documento HTML. Las pseudo-clases son abstracciones de estados concretos de un elemento que de otro modo serían inaccesibles.

La **pseudo-clase** se sitúa a continuación del elemento separado por "**:**"
a**:hover**

Pseudo-clases **:link, :hover y :visited**

```
a:link {
    color: #FE0;
    background-color: black;
    text-decoration:none;
    }
```

```
a:hover {
    color: black;
    background-color: #FE0;
    text-decoration:none;
    }
```

```
a:visited {
    color: #444;
    background-color: black;
    text-decoration:none;
    }
```

a:link > re-define el estilo por defecto del texto contenido en el elemento **a**.
a:hover > Indica el estilo que tendrán los enlaces al pasar el *mouse* sobre ellos.
a:visited > Creamos un estilo para indicar cómo se van a mostrar los enlaces que ya han sido visitados.

Pseudo-clase **:first-child**

```
ul li:first-child {
    background-color: red;
    }
```

Mediante este selector podemos hacer referencia siempre al primer hijo de otro elemento que lo contenga. De esta forma seleccionaremos el primer **<li>** del **<ul>**.

Pseudo-elementos **:first-line y :first-letter**

```
p:first-line {
    background-color: red;
    font-weight:bold;
    }
```

selección de la primera línea de <p>

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

selección de la primera letra de <p>

Un **pseudo-elemento** permite aplicar un estilo **a sólo una parte** del elemento seleccionado en lugar de hacerlo a todo el elemento en sí.

Pseudo-elementos **:before y :after**

```
ul li:before {
    content: url("bullet.png");
    padding-left: 10px;
    }
```

**insertamos una imagen antes del **

```
div#content:after {
    content: "Final del artículo";
    font-weight: bold;
    }
```

insertamos un texto después de #content

Los pseudo-elementos **:before** y **:after** nos permiten añadir contenido antes o después de un elemento determinado y especificar el formato en el que se mostrará dicho contenido.

Consideraciones a tener en cuenta para escribir código CSS limpio

· Estructura bien el código HTML.
· Utiliza nombres de clases identificativos, no descriptivos:

Incorrecto: ~~<h1 class=textogrande1>Título</h1>~~ | **Correcto**: <h1 class=encabezado>Título</h1>

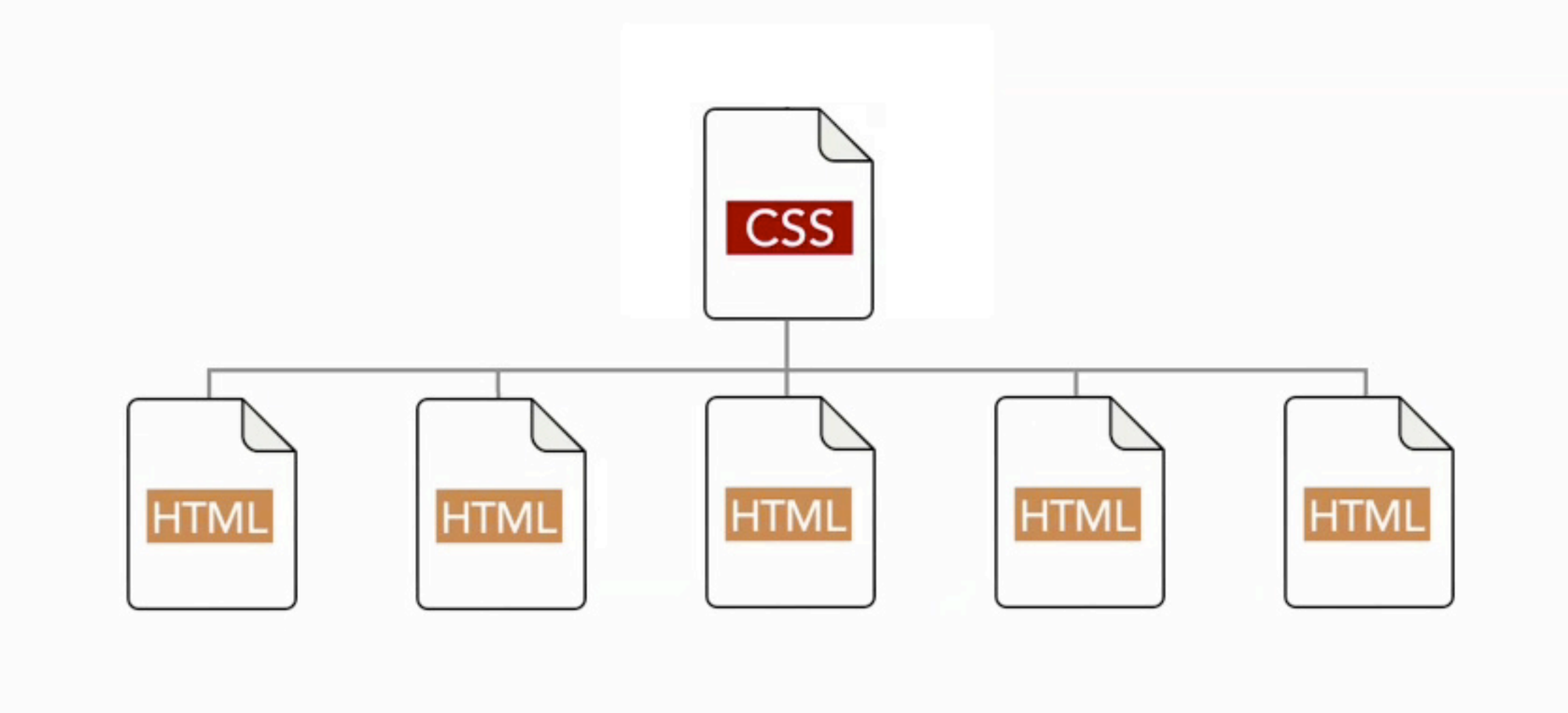
· Intenta minimizar el código HTML para evitar generar CSS redundante.
· Evita duplicar estilos mediante la combinación de selectores CSS y su especificidad.
· Añade comentarios a bloques de código para mejorar la legibilidad y la depuración de errores.

Aplicando hojas de estilo

Podemos adjuntar hojas de estilo a documentos HTML
de 4 formas diferentes.
Veamos las ventajas e inconvenientes de cada una de estas formas:

Hojas de estilo **Externas**

Como vimos anteriormente, este sistema nos permite vincular una misma hoja de estilos a todas las páginas de nuestra web



Hojas de estilo **Externas**

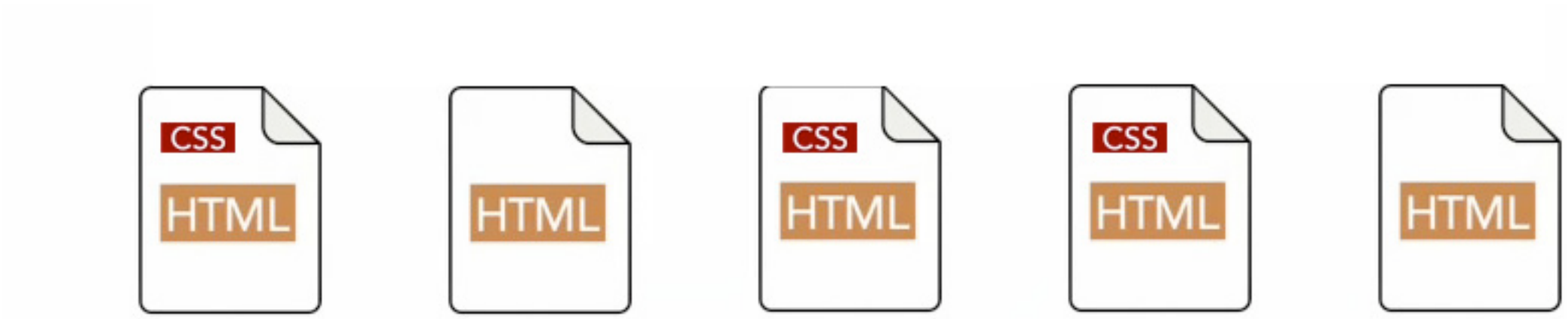
Creamos un archivo con extensión **.css** que contiene todas las declaraciones de formato y lo vinculamos al encabezado de cada una de las páginas de nuestra web:

```
<head>
   <link type="text/css" href="estilos.css" rel="stylesheet" media="screen">
</head>
```

De esta forma, modificando un estilo en este archivo afectamos al look&feel de toda la web.

Hojas de estilo **Internas**

Las hojas de estilo internas están incluidas directamente en el encabezado del documento HTML.



Hojas de estilo **Internas**

A pesar de que este sistema es menos versátil que el anterior, es perfecto para sobreescribir un estilo concreto en una página específica.

```
<head>
   <style>
      body {font-family:Helvetica, Arial, San-serif;}
      p{font-size:0.9em;}
   </style>
</head>
```

Hojas de estilo **en línea**

Se trata de estilos que se aplican directamente a los elementos HTML como atributos.

```
<body>
   <div>
      <p style=”font-size:2; color:#0FF;”>
            ... contenido ...
      </p>
   </div>
</body>
```

Hojas de estilo **en línea**

La única aplicación útil a día de hoy es en **E-mail marketing** basados en HTML ya que no pueden contener enlaces a archivos **.css** externos.

```
<table>
   <tr>
      <td style=”font-size:2; color:#0FF;”>
            ... contenido ...
      </td>
   </tr>
</table>
```

Hojas de estilo **importadas**

Podemos importar una hoja de estilos externa a nuestro documento HTML mediante el identificador **@** seguido de **import** y especificar el medio en el que ha de aplicarse:

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

Cascada, Herencia, Especificidad y Acumulativo.

Hojas de estilo en cascada

Se dice que las hojas de estilo son en cascada por el orden en el que las aplica el navegador.

1. **Externas**: <**link** type=”text/css” href=”estilos.css” rel=”stylesheet” >
2. **Internas**: <**style**>body {font-family:Helvetica, Arial, San-serif;}</**style**>
3. **En línea**: <p **style**=”font-size:2; color:#0FF;”>... contenido ...</p>

Hojas de estilo en cascada

Conflicto entre estilos

En el caso de conflicto entre dos estilos con el mismo nivel de especificidad: **prevalecerá el último estilo aplicado**.

Cuando se trata de estilos en el mismo documento, ya sea una hoja de estilos interna, externa o incluso estilos dentro de la misma declaración: **prevalecerá el último estilo aplicado**

Hojas de estilo en cascada

Conflicto entre estilos basado en el orden de aplicación

En la siguiente declaración, el color del elemento “p” será rojo:

```
p {
    font-size: 1em;
    color: black; --------> inicialmente aplicamos el color negro.
    margin:10px;
    color: red;  --------> prevalecerá el último estilo aplicado
    }
```

Herencia

Estilos heredados

La herencia de estilos sucede cuando un elemento “hijo” hereda un estilo aplicado a otro elemento “padre”.

Se dice que un elemento es “hijo” **cuando está contenido** dentro de otro elemento. A ese elemento contenedor se le conoce como “padre”.

Herencia

Estilos heredados

El concepto de Herencia de estilos nos permite escribir código extremadamente eficiente ya que nos evita tener que especificar la misma propiedad para cada uno de los elementos.

```
h1  {font-size: 1em; color: #000;}
h2  {font-size: 1em; color: #000;}
p  {font-size: 1em; color: #000;}
li  {font-size: 1em; color: #000;}
```

Incorrecto

```
body  {font-size: 1em; color: #000;}
```

Correcto

Especificidad

Selectores con mayor preponderancia

La especificidad define que selector es más específico para determinada situación.

1. **ID**: Son los selectores más específicos.
2. **Clases**:
3. **Etiquetas**: los menos específicos.
4. **Combinaciones**: La combinación de selectores que utilice **ID + etiqueta** será más específica que la que sólo utilice **etiqueta + etiqueta**.

Acumulativo

Los estilos se suman

El concepto Acumulativo define que un elemento se mostrará con todos los estilos que le afecten tanto si proviene de una hoja de estilos externa, una herencia, un valor por defecto del navegador web o un estilo en línea.

En caso de conflicto entre estilos:
prevalecerá el más específico y el último estilo aplicado.

Disposición de elementos

mediante estilos CSS

Disposición de elementos

Mediante las reglas de formato CSS podemos definir como se van a distribuir los elementos HTML en pantalla, donde se van a colocar y cual va a ser su punto de referencia.

Hay 2 tipos de elementos:
En línea y de Bloque

Disposición de elementos

Elementos en Línea

Se dice que un elemento HTML es en Línea cuando al renderizarse en el navegador sigue el **Flujo normal de la página**.

Ejemplos de elementos en Línea:

**<a> <img>   <b> <i> **

Disposición de elementos

Elementos en Bloque

Se dice que un elemento HTML es en Bloque cuando al renderizarse en el navegador no permite que otros elementos se posicionen a su lado. Por norma general ocupan el 100% del ancho de pantalla.

Ejemplos de elementos en Bloque:

<p> <h1> <hr>   <form> <table>

Disposición de elementos

La propiedad: Display

Hay veces en las que necesitamos que un elemento de Bloque se comporte en la página como uno en Línea y viceversa.
Mediante la propiedad **Display** podemos hacer precisamente eso:

```
p {
    display:inline;
    }
```

Cambio de Bloque a Línea

```
img {
    display:block;
    }
```

Cambio de Línea a Bloque

```
img {
    display:none;
    }
```

Ocultamos en elemento

Disposición de elementos

Disponemos de 3 formas de distribuir
los elementos HTML en pantalla:

Flujo normal de página
Elementos posicionados
Elementos flotados

Disposición de elementos

Flujo normal de la página

Si no aplicamos ninguna regla de formato a un documento HTML el **Flujo normal de la página** sería la forma en la que los elementos se distribuirían en pantalla.

Disposición de elementos

Flujo normal de la página

Documento HTML

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec ut turpis in ligula pretium dictum. Sed imperdie.

Vestibulum tristique

Viverra enim, at lobortis odio congue ut. **Mauris a leo** a nisi imperdiet fringilla vitae pellentesque diam. Nam hendrerit commodo nibh non elementum.

En este ejemplo, los elementos de **Bloque <p>** y **<h1>** se distribuyen uno debajo del otro siguiendo el flujo normal de la página.

El elemento en **Línea ** sigue el flujo de la página dentro del elemento de bloque que lo contiene.

Disposición de elementos

Elementos posicionados

Cuando queremos alterar el comportamiento normal de un elemento en el flujo de la página utilizamos el atributo: **Position** junto alguno de sus 5 valores posibles:

static
relative
absolute
fixed
inherit

Disposición de elementos

position:static

Hace referencia al valor por defecto. Indica al elemento HTML que se posicione utilizando el flujo normal de la página.

```
div {
    position:static;
    }
```

Disposición de elementos

position:relative

El elemento sigue dentro del flujo normal de la página pero ahora podemos definir una distancia relativa a su posición original mediante los atributos:
top, **left**, **right** y **bottom**.

```
div {
    position:relative;
    top: 20px;
    left: 50px;
    }
```

Disposición de elementos

position:relative

Este tipo de posicionamiento no suele utilizarse con sus atributos **top**, **left**, **right** y **bottom** ya que al seguir dentro del flujo normal de la página no aporta un resultado útil.

Por norma general se aplica a elementos que contienen a su vez otros elementos (hijos) posicionados de forma **absoluta**.

Disposición de elementos

position:absolute

"Saltamos" al elemento del flujo normal de la página y lo posicionamos a una distancia específica mediante los atributos:
top, **left**, **right** y **bottom**.

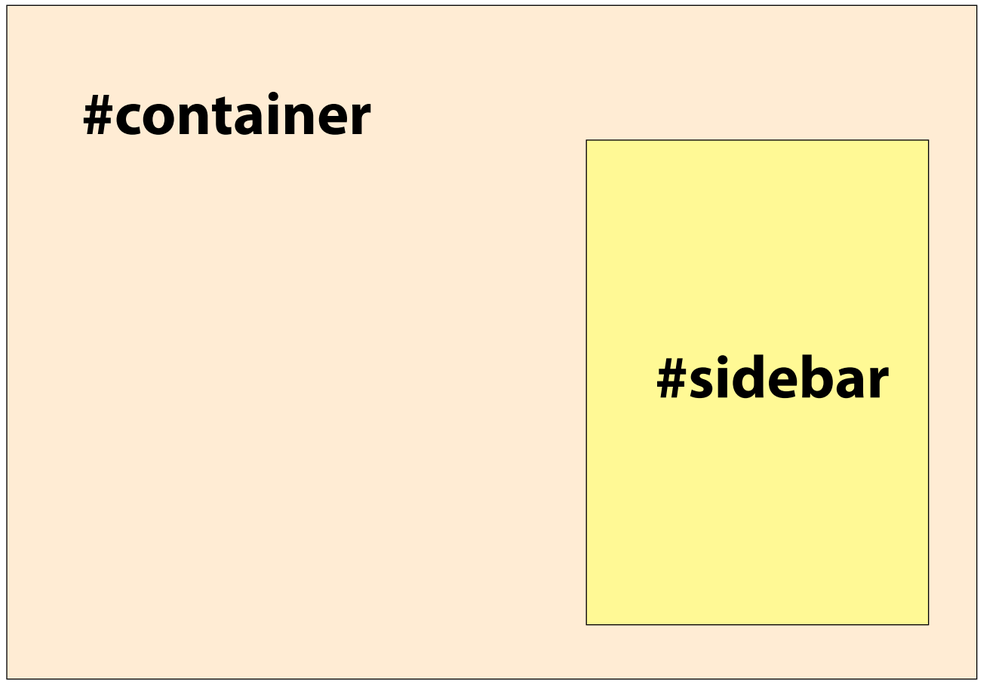
La referencia será el elemento padre posicionado mediante Position o en su defecto la etiqueta <body>

```
div {
    position:absolute;
    bottom: 30px;
    right: 30px;
    }
```

Disposición de elementos

position:absolute

Ejemplo práctico:




```
#container {
    position:relative;
    }

#sidebar {
    position:absolute;
    top:50px;
    right: 20px;
    }
```

Disposición de elementos

position:fixed

Un elemento posicionado mediante el atributo fixed toma como referencia la ventana del navegador y permanece *fijado* en pantalla incluso cuando el usuario hace *scroll*.

Estos elementos tampoco siguen el flujo normal de la página.

Disposición de elementos

position:inherit

El término **inherit** significa que el elemento heredará el valor del atributo de su elemento padre. De esta forma podemos transmitir a elementos hijos atributos que de otra forma no serían heredados.

Disposición de elementos

z-index

Cuando posicionamos elementos en pantalla puede darse el caso de que se solapen entre sí. Por norma general, el navegador sigue el orden de apilamiento según aparecen en el código HTML.

Mediante la propiedad **z-index** podemos reordenar dicho orden por defecto.

Disposición de elementos

z-index

La propiedad **z-index** sólo se aplica a elementos HTML posicionados. A mayor sea el valor numérico de dicho índice más cerca del usuario quedará el contenido. El valor "**auto**" hereda el índice de su contenedor *padre*:

```
#bottom {
    x-index:1;
    }
```

Quedará debajo de #top

```
#top {
    x-index:1;
    }
```

Quedará sobre de #bottom

```
#top {
    x-index:auto;
    }
```

Heredará el valor de z-index

Disposición de elementos

Elementos flotados

Un elemento flotado no se encuentra en el flujo normal de la página.

Podemos indicar que la caja flotada se sitúe a la izquierda, derecha o no esté flotada mediante los atributos: **left**, **right** y **none**.

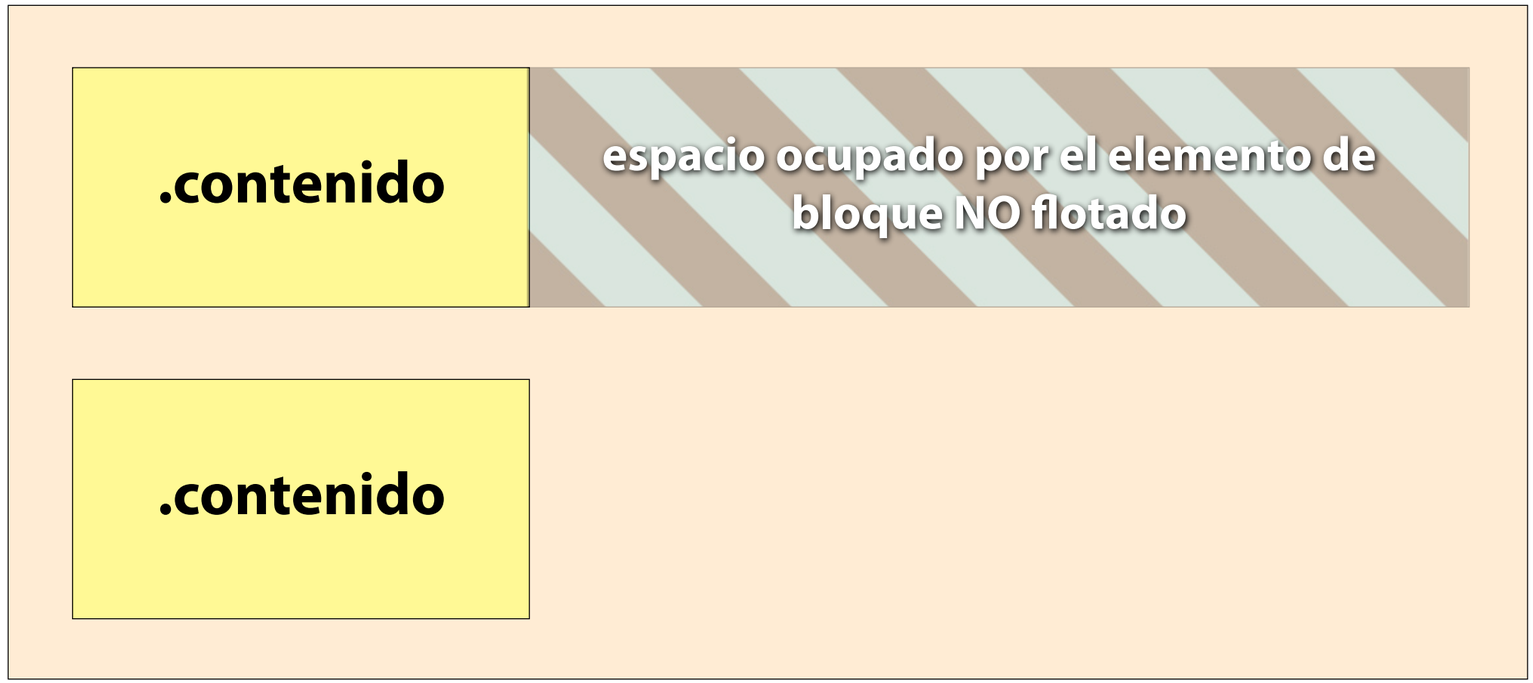
```
div {
    float:left;
    }

div {
    float:right;
    }
```

Disposición de elementos

Elementos flotados

Ejemplo práctico: Elementos de bloque **NO flotados** de 100px de ancho:

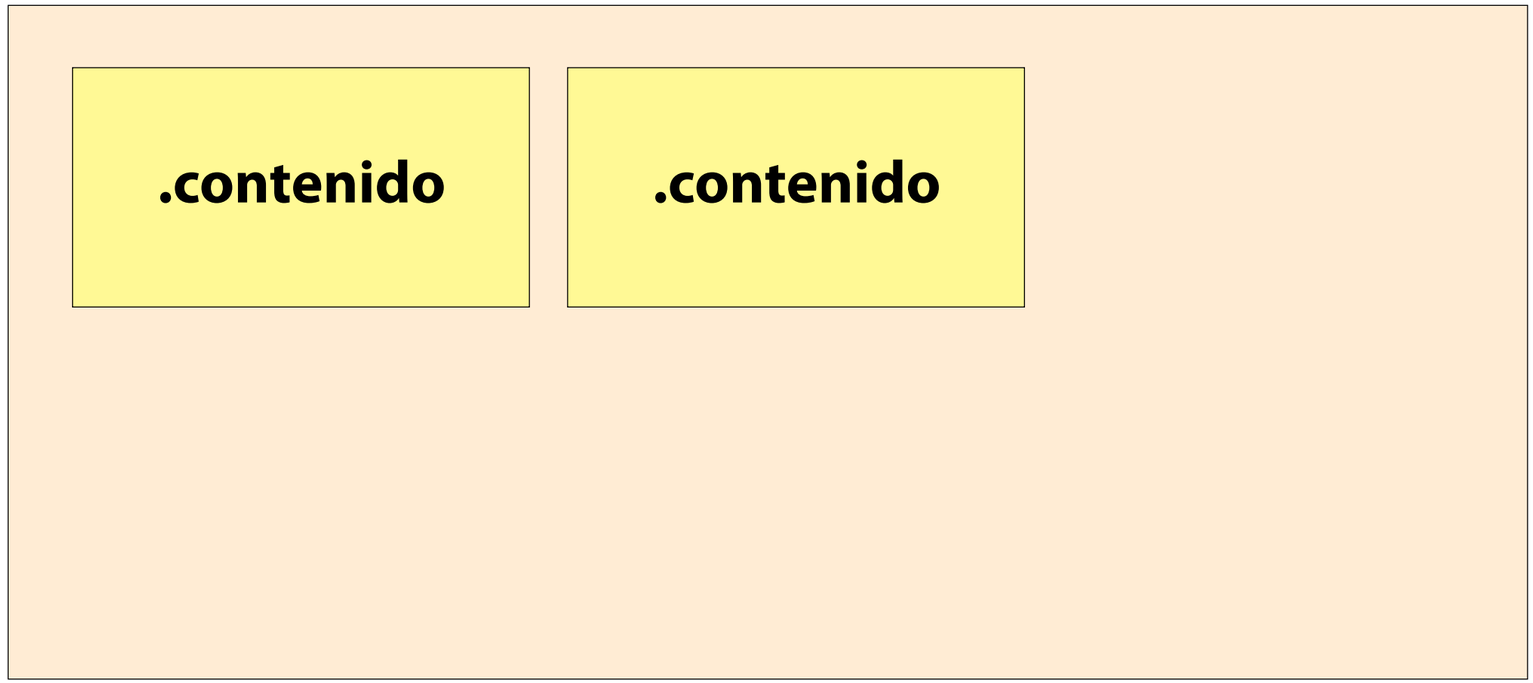



```
.contenido {
    width: 100px;
}
```

Disposición de elementos

Elementos flotados

Ejemplo práctico: Elementos de bloque flotados a la izquierda:

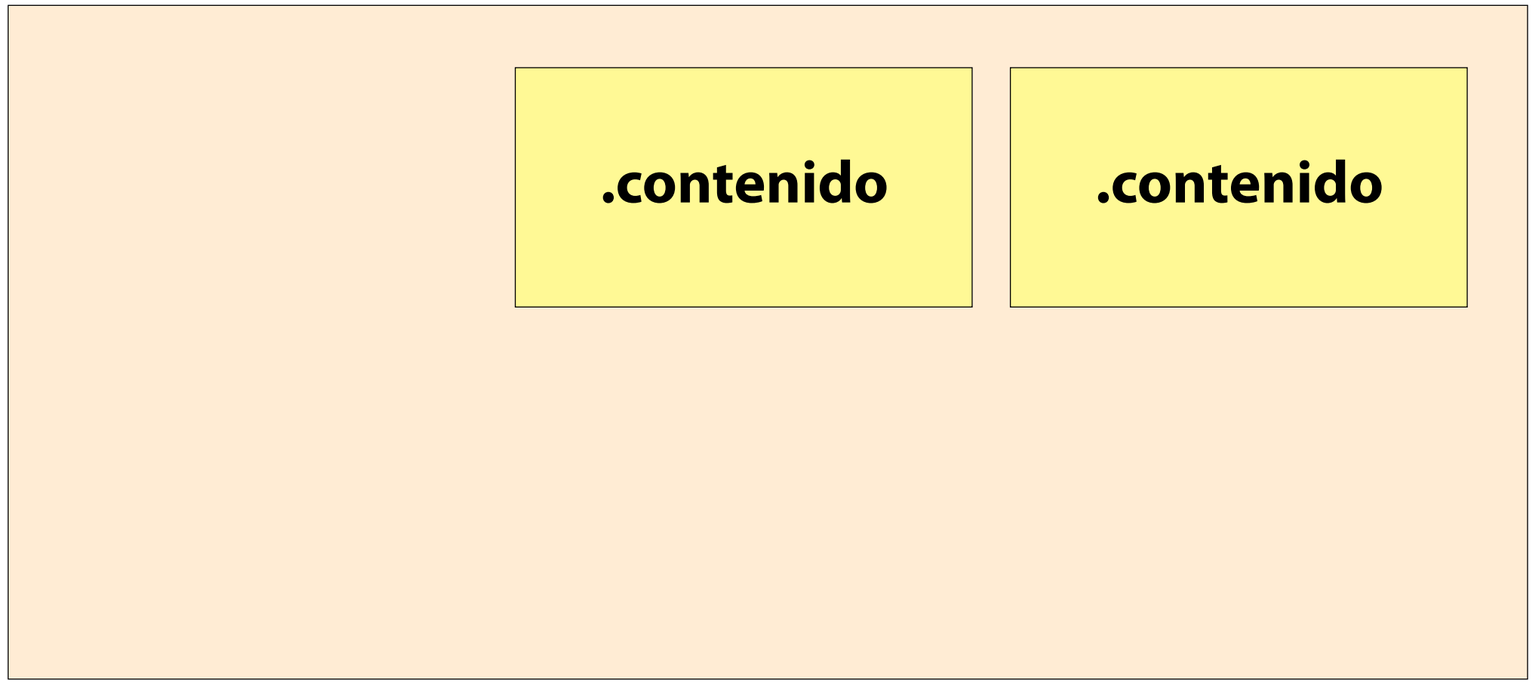



```
.contenido {
    float:left;
    width: 100px;
    }
```

Disposición de elementos

Elementos flotados

Ejemplo práctico: Elementos de bloque flotados a la derecha:




```
.contenido {
    float:right;
    width: 100px;
    }
```

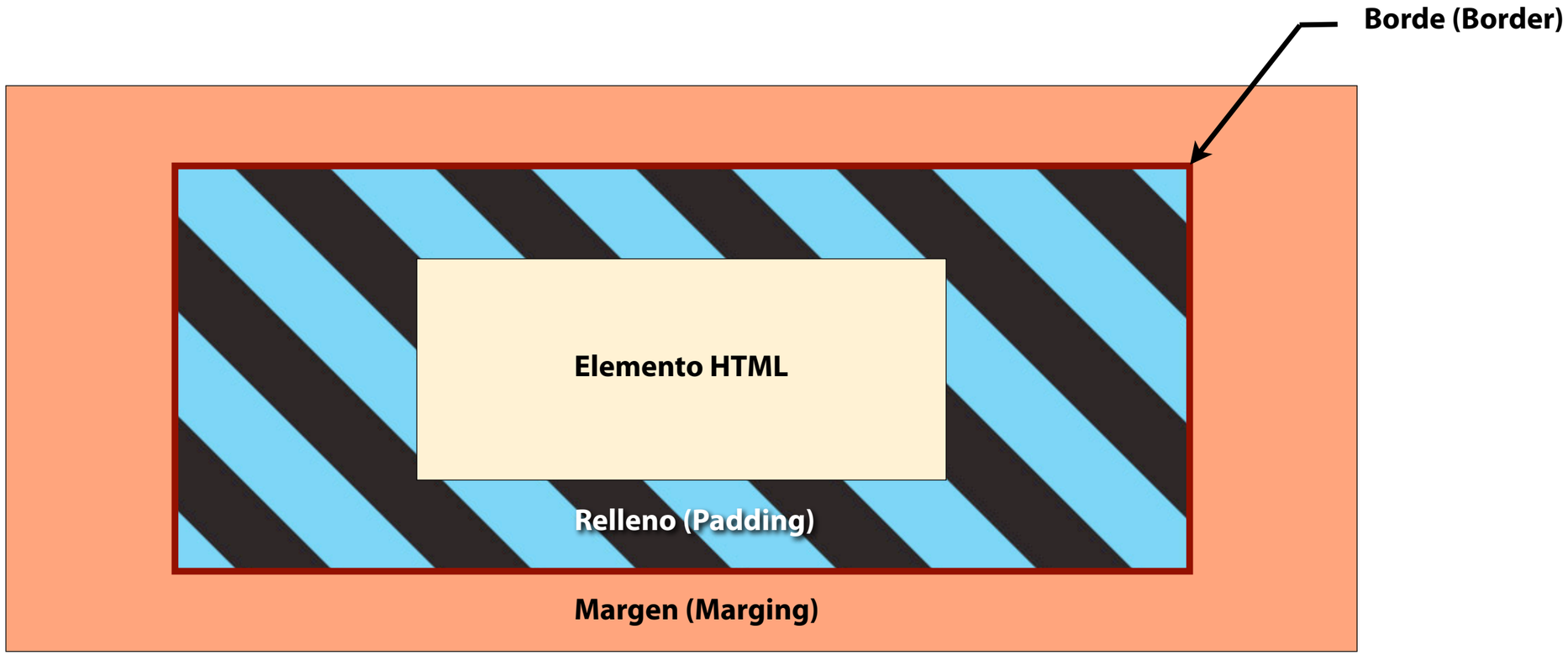
El Box Model

Composición mediante el modelo de cajas

Cuando el navegador renderiza los elementos de una página HTML los interpreta como si fueran *cajas*. Como hemos visto en el capítulo anterior, algunos de estos elementos siguen el flujo de la página y otros no pero todos esllos tienen los siguientes atributos de caja:

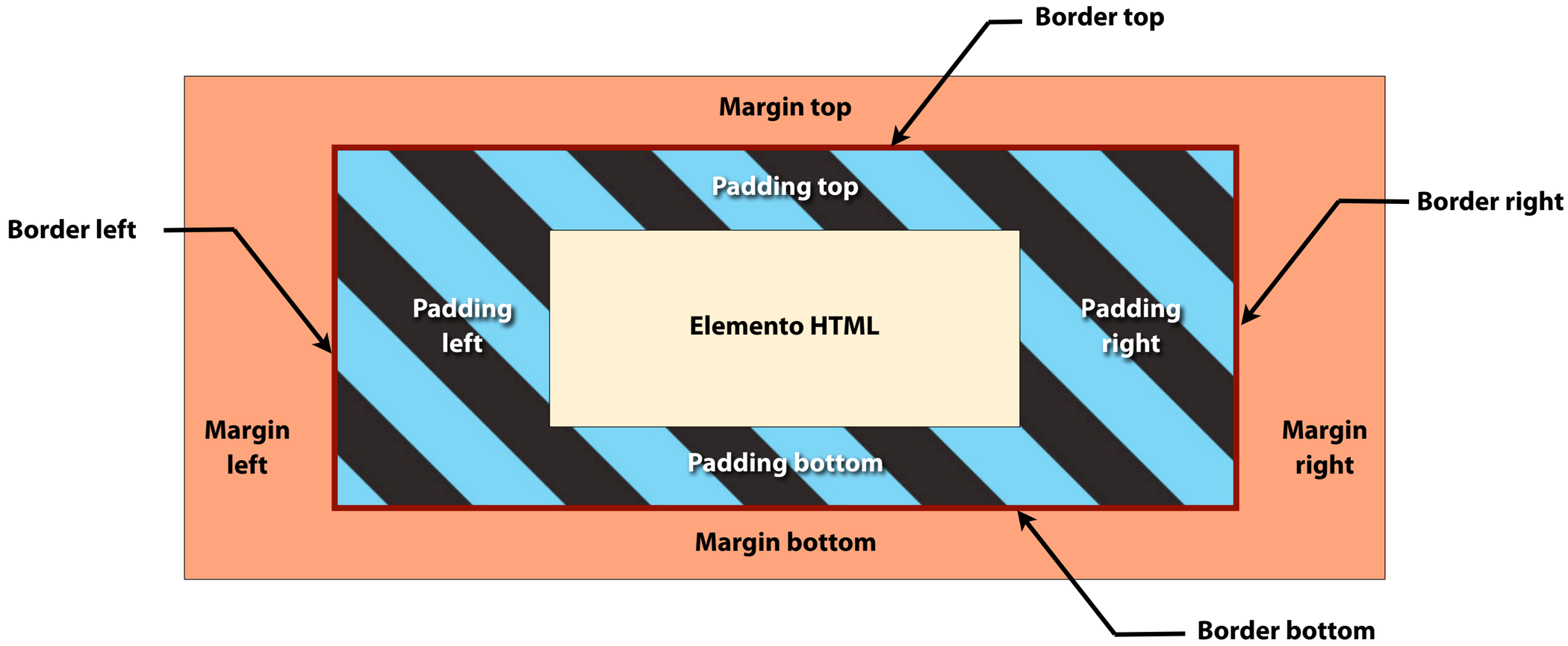
Box Model

Elementos de la caja



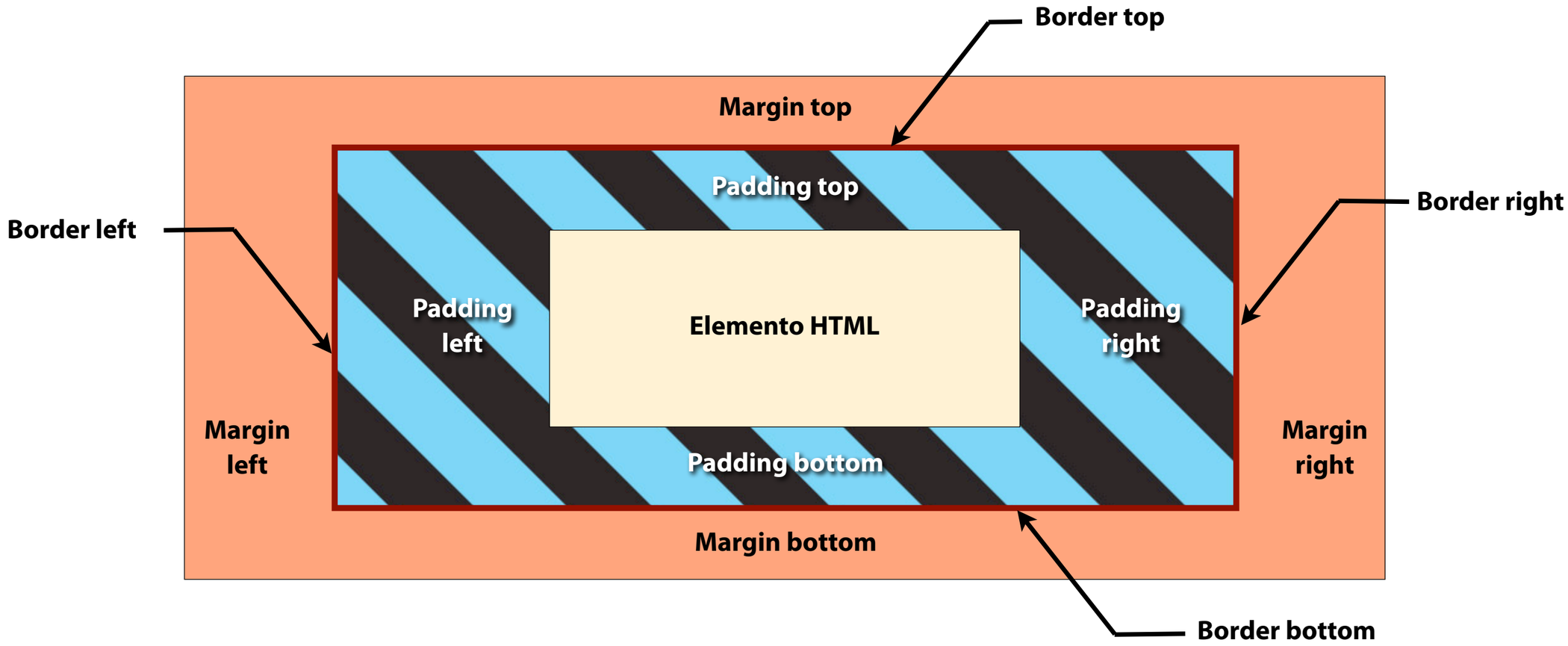
Box Model

Cada elemento tiene unas **áreas de contenido** divididas en **top, right, bottom, left**



Box Model

Cada elemento tiene unas **áreas de contenido** divididas en **top, right, bottom, left**



Box Model

Sintaxis de los atributos padding border y margin

Los atributos del **box model** se pueden aplicar:

· Independientemente a cada uno de sus límites: top, right, bottom y left.

· Combinando top-bottom y left-right.

· Aplicando a todos el mismo valor.

```
p {
    padding-top:10px;
    padding-left:20px;
}
```

Valores independientes a cada límite

```
p {
    border:10px solid black;
}
```

La propiedad Border requiere : “width”, “style” y “color”

```
p {
    margin:10px;
}
```

todos los límites a 10px

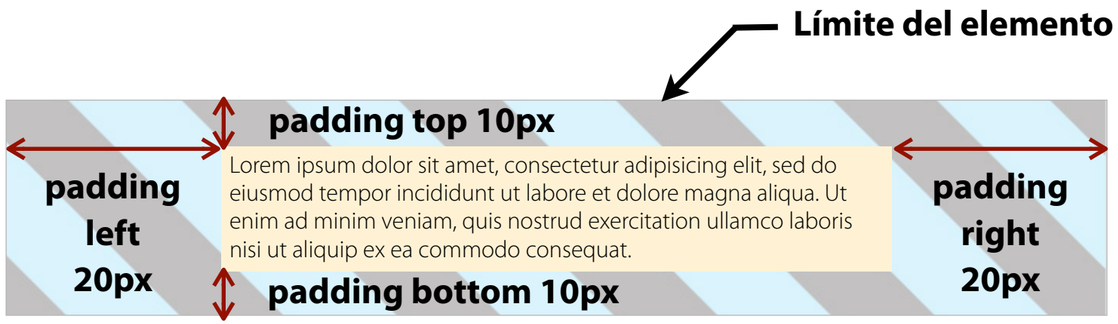
Box Model

padding

El **padding** establece un espacio de relleno entre el límite del elemento y su contenido. El valor final de alto/ancho del elemento será el valor resultante de la suma de su altura/anchura + padding.

```
p {
    width: 50px;
    padding:10px 20px;
    }
```

padding genérico



Box Model

border

El **border** indica un grosor para el límite del elemento al que se aplica. El valor final de alto/ancho del elemento será el valor resultante de la suma de su altura/anchura + padding + border.

```
p {
    border-left-width:4px;
    border-left-color:black;
    border-left-style:solid;
    }
```

sintaxis completa (left)

```
p {
    border-left: solid black 4px;
    }
```

sintaxis resumida

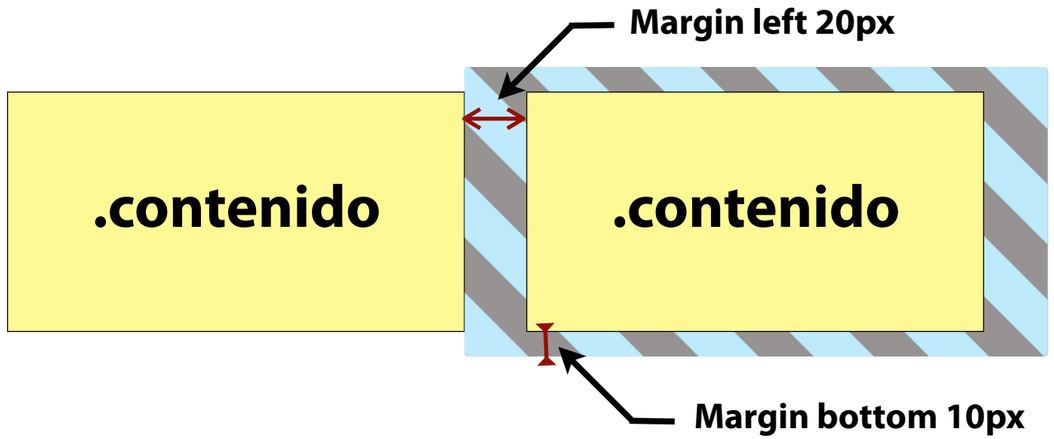
Box Model

margin

El **margin** añade un espacio entre el elemento al que aplicamos la regla CSS y los que se encuentran a su alrededor.

```
.contenido {
    width: 150px;
    margin:10px 20px;
    }
```

margin top-bottom 10px y left-right 20px

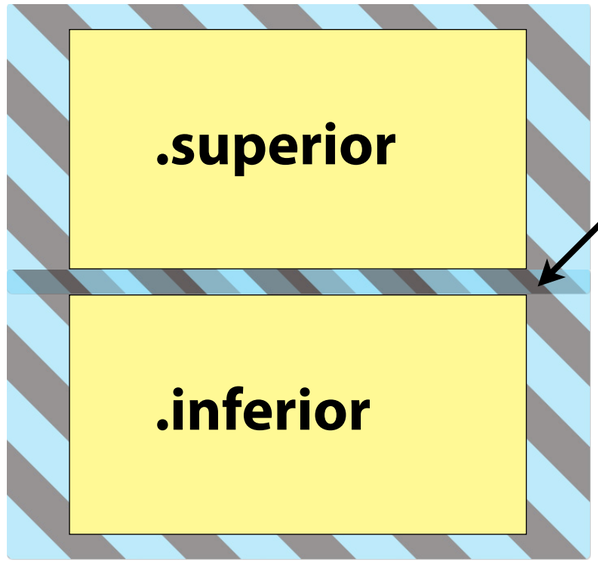


Box Model

vertical margin

Una característica a tener en cuenta respecto a **margin** es que los márgenes verticales **no se suman**.

Prevalece el *margin* que tenga un valor más alto.



El valor de margen inferior de la capa .superior y el valor superior de .inferior colapsan quedando únicamente uno de los dos valores.