

# CURSO .CSS

## INTERMEDIO



Autor: Jon Vadillo  
[www.jonvadillo.com](http://www.jonvadillo.com)

# Contenidos

- Selectores CSS
- Pseudo clases
- Especificidad
- Herencia
- Propiedades abreviadas
- Page Layout
  - Display
  - Float

## Selectores de ID y de clase

Permiten dar estilos diferentes a un mismo tipo de elemento en función a su ID o clase.

# Sintaxis

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

```
#id {  
  color: blue;  
}
```

# Referenciar desde HTML

Podemos aplicar los estilos definidos con selectores de tipo clase o ID utilizando los atributos `id` y `class`.

En el selector de clase también se puede especificar que solo se aplique a un tipo de elementos concreto:

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

```
<div id="top">  
  <h1>Chocolate curry</h1>  
  <p class="intro">Introducción</p>  
  <p class="contenido">Contenido</p>  
</div>
```

# Ejercicio 2

## Aprendiendo CSS

### Lorem ipsum dolor sit amet consectetur

Lorem ipsum dolor sit amet consectetur adipiscing elit ad **pretium consequat** orci, ridiculus aenean mus faucibus **mauris quam** est sociis cursus lacinia. Malesuada pellentesque laoreet nascetur mauris mus, accumsan **habitasse** potenti nec libero, duis sapien viverra imperdiet.

Placerat ullamcorper parturient dui iaculis porttitor **malesuada cum sociosqu consequat** ultricies, mi scelerisque platea eu facilisi urna curabitur **justo** eleifend quam.

Magnis ante inceptos euismod potenti etiam enim non vehicula.

### Lorem ipsum dolor sit amet consectetur

Aliquet magna **nostra duis facilisis risus** suscipit cum scelerisque, turpis urna at laoreet luctus pretium mi, semper sociis sapien mus parturient ac justo.

At elementum penatibus etiam **justo** semper blandit pretium **turpis nostra**, euismod pharetra congue porttitor fusce ad per vestibulum class, torquent aenean ut **facilisis** eros nec ante sagittis. At iaculis curae aliquam **vitae sem id semper** dictum mus quisque, vivamus pharetra molestie praesent vestibulum nisl interdum congue dictumst penatibus urna, enim magnis porta a auctor non gravida aliquet sollicitudin.

### Lorem ipsum dolor sit amet consectetur

Primis mattis vivamus augue rhoncus dictum nunc fermentum cras bibendum magnis leo risus duis ridiculus phasellus, massa sollicitudin hac posuere vitae consequat nostra quis pulvinar



## Aprendiendo CSS

### Lorem ipsum dolor sit amet consectetur

Lorem ipsum dolor sit amet consectetur adipiscing elit ad **pretium consequat** orci, ridiculus aenean mus faucibus **mauris quam** est sociis cursus lacinia. Malesuada pellentesque laoreet nascetur mauris mus, accumsan **habitasse** potenti nec libero, duis sapien viverra imperdiet.

Placerat ullamcorper parturient dui iaculis porttitor **malesuada cum sociosqu consequat** ultricies, mi scelerisque platea eu facilisi urna curabitur **justo** eleifend quam.

Magnis ante inceptos euismod potenti etiam enim non vehicula.

### Lorem ipsum dolor sit amet consectetur

Aliquet magna **nostra duis facilisis risus** suscipit cum scelerisque, turpis urna at laoreet luctus pretium mi, semper sociis sapien mus parturient ac justo.

At elementum penatibus etiam **justo** semper blandit pretium **turpis nostra**, euismod pharetra congue porttitor fusce ad per vestibulum class, torquent aenean ut **facilisis** eros nec ante sagittis. At iaculis curae aliquam **vitae sem id semper** dictum mus quisque, vivamus pharetra molestie praesent vestibulum nisl interdum congue dictumst penatibus urna, enim magnis porta a auctor non gravida aliquet sollicitudin.

### Lorem ipsum dolor sit amet consectetur

Primis mattis vivamus augue rhoncus dictum nunc fermentum cras bibendum magnis leo risus duis ridiculus phasellus, massa sollicitudin hac posuere vitae consequat nostra quis pulvinar odio ultrices quam commodo.

Phasellus dictum bibendum lacus praesent ultricies felis facilisi potenti mus ultrices, sociosqu nunc pharetra aenean conubia maecenas **turpis nostra** habitant tellus dis velit, cursus varius mattis suscipit ridiculus blandit natoque ad faucibus.

# Selectores basados en relaciones

- Selectores de elementos descendientes
- Selectores de elementos hijos
- Selectores del primer hijo
- Selectores de elementos adyacentes

# Selectores basados en relaciones

$A E$

Cualquier elemento E que es un **descendiente** de un elemento A (que es: un hijo o un hijo de un hijo etc.)

$A > E$

Cualquier elemento E que es un **hijo** de un elemento A

$E: \text{first-child}$

Cualquier elemento E que es el **primer hijo** de su padre

$B + E$

Cualquier elemento E que es el **siguiente hermano** de un elemento B (es decir: el próximo hijo del mismo padre)



# Ejercicio 3

## [HTML & CSS: Curso práctico avanzado](#)

Aunque los inicios de [Internet](#) se remontan a los años sesenta, no ha sido hasta los años noventa cuando, gracias a la Web, se ha extendido su uso por todo el mundo. En pocos años, la Web ha evolucionado enormemente: se ha pasado de páginas sencillas, con pocas imágenes y contenidos estáticos que eran visitadas por unos pocos usuarios a *páginas complejas, con contenidos dinámicos que provienen de bases de datos y que son visitadas por miles de usuarios al mismo tiempo*.

Todas las páginas están internamente construidas con la misma tecnología, con el *Lenguaje de marcas de hipertexto* (Hypertext Markup Language, [HTML](#)) y con las *Hojas de estilo en cascada* (Cascading Style Sheets, [CSS](#)).

Este libro es *adecuado para cualquiera que tenga interés en aprender a desarrollar sus propias páginas web*. No son necesarios conocimientos previos para aprender con este libro, lo único que es necesario es saber utilizar un ordenador y saber navegar por la Web.



## [HTML & CSS: Curso práctico avanzado](#)

Aunque los inicios de [Internet](#) se remontan a los años sesenta, no ha sido hasta los años noventa cuando, gracias a la Web, se ha extendido su uso por todo el mundo. En pocos años, la Web ha evolucionado enormemente: se ha pasado de páginas sencillas, con pocas imágenes y contenidos estáticos que eran visitadas por unos pocos usuarios a *páginas complejas, con contenidos dinámicos que provienen de bases de datos y que son visitadas por miles de usuarios al mismo tiempo*.

Todas las páginas están internamente construidas con la misma tecnología, con el *Lenguaje de marcas de hipertexto* (Hypertext Markup Language, [HTML](#)) y con las *Hojas de estilo en cascada* (Cascading Style Sheets, [CSS](#)).

Este libro es *adecuado para cualquiera que tenga interés en aprender a desarrollar sus propias páginas web*. No son necesarios conocimientos previos para aprender con este libro, lo único que es necesario es saber utilizar un ordenador y saber navegar por la Web.

# Selector de atributo

Permiten seleccionar elementos con un atributo concreto o con un valor determinado para un atributo.

`[nombre_atributo]`

Cualquier elemento con ese atributo.

`[nombre_atributo=valor]`

Cualquier elemento con el valor indicado para el atributo.

```
/* Se muestran de color azul todos los
enlaces que tengan
    un atributo "class",
independientemente de su valor */
a[class] { color: #00f; }

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

/* Se muestran de color azul todos los
enlaces que apunten
    al sitio "http://www.ejemplo.com" */
a[href="http://www.google.com"] {
    color: #f000;
}
```

Todos los selectores de atributo en:

[https://developer.mozilla.org/es/docs/Web/CSS/Selectores\\_atributo](https://developer.mozilla.org/es/docs/Web/CSS/Selectores_atributo)

## Ejercicio 4

- Crea un formulario formado por elementos de tipo: text, radio, checkbox, select, email y textarea.
- Aplica estilos (color de fuente, color de fondo, etc) en función del atributo “type” o del tipo de elemento.

# Pseudoclases

Sirven para especificar un estado especial del elemento seleccionado.

```
/* Selecciona cualquier elemento <a>
cuando está "mantenido (hovered)" */
a:hover {
    color: orange;
}
/* Selecciona cualquier <input>
deshabilitado */
input:disabled {
    background: #ccc;
}
/* Selecciona cualquier <input>
requerido */
input:required {
    border: 1px dashed red;
}
/* Coincide con cualquier
checked/selected radio, checkbox, u
option */
:checked {
    margin-left: 25px;
    border: 1px solid blue;
}
/* Elemento Checkbox, cuando está
marcado */
input[type="checkbox"]:checked{
    box-shadow: 0 0 0 3px orange;
}
```

# Sintaxis de pseudoclases

```
selector:pseudoclase {
    propiedad: valor;
}
```

Todos los selectores de atributo en:

[https://developer.mozilla.org/es/docs/Web/CSS/Selectores\\_atributo](https://developer.mozilla.org/es/docs/Web/CSS/Selectores_atributo)

# Listado de pseudoclases

<code>:active</code>	<code>:indeterminate</code>	<code>:only-child</code>
<code>:checked</code>	<code>:in-range</code>	<code>:only-of-type</code>
<code>:default</code>	<code>:invalid</code>	<code>:optional</code>
<code>:dir()</code>	<code>:lang()</code>	<code>:out-of-range</code>
<code>:disabled</code>	<code>:last-child</code>	<code>:read-only</code>
<code>:empty</code>	<code>:last-of-type</code>	<code>:read-write</code>
<code>:enabled</code>	<code>:left</code>	<code>:required</code>
<code>:first</code>	<code>:link</code>	<code>:right</code>
<code>:first-child</code>	<code>:not()</code>	<code>:root</code>
<code>:first-of-type</code>	<code>:nth-child()</code>	<code>:scope</code>
<code>:fullscreen</code>	<code>:nth-last-child()</code>	<code>:target</code>
<code>:focus</code>	<code>:nth-last-of-type()</code>	<code>:valid</code>
<code>:hover</code>	<code>:nth-of-type()</code>	<code>:visited</code>

## Ejercicio 5

- Crea un listado con 5 hipervínculos a páginas distintas. Aplica estilos CSS siguiendo los criterios descritos a continuación:
  - Cada enlace tiene un color de texto en función de la dirección (url).
  - Al poner el ratón sobre ellos, todos muestran un borde negro y fondo blanco.
  - Una vez visitados, tienen un fondo de color #dbdbdb

## Ejercicio 6













- Modifica el formulario del Ejercicio 4 y añade los estilos necesarios para que:
  - Los checkbox marcados por el usuario (checked) tiene que cambiar de color.
  - Al poner el foco en un elemento, este deberá cambiar el color de fondo.
  - Los elementos deshabilitados tendrán fondo de color negro (#000).
  - Añade un estilo al pasar el ratón sobre cada elemento.



## Más específico = Más prioridad

Si un elemento tiene más de una regla, CSS da mayor prioridad a la regla que tiene el selector más específico.

Un selector ID es más específico que un selector de class, que a su vez es más específico que un selector de etiquetas.

 <b>a</b> 1 x element selector Sith power: 0,0,1	 <b>p a</b> 2 x element selectors Sith power: 0,0,2	 <b>.foo</b> 1 x class selector * Sith power: 0,1,0	 <b>a.foo</b> 1 x element selector 1 x class selector Sith power: 0,1,1
 <b>p a.foo</b> 2 x element selectors 1 x class selector Sith power: 0,1,2	 <b>.foo .bar</b> 2 x class selectors Sith power: 0,2,0	 <b>p.foo a.bar</b> 2 x element selectors 2 x class selectors Sith power: 0,2,2	 <b>#foo</b> 1 x id selector Sith power: 1,0,0
 <b>a#foo</b> 1 x element selector 1 x id selector Sith power: 1,0,1	 <b>.foo a#bar</b> 1 x element selector 1 x class selector 1 x id selector Sith power: 1,1,1	 <b>.foo .foo #foo</b> 2 x class selectors 1 x id selector Sith power: 1,2,0	 <b>style</b> 1 x style attribute Sith power: 1,0,0,0



\* Same specificity  
 class selector =  
 attribute attribute =  
 pseudo-classes



!important

- **p** tiene especificidad de 1 (1 HTML selector)
- **div p** tiene especificidad de 2 (2 HTML selectors, 1+1)
- **.tree** tiene especificidad de 10 (1 class selector)
- **div p.tree** tiene especificidad de 12 (2 HTML selectors + a class selector, 1+1+10)
- **#baobab** tiene especificidad de 100 (1 id selector)
- **body #content .alternative p** tiene especificidad de 112 (HTML selector + id selector + class selector + HTML selector, 1+100+10+1)

```
/* specificity: 0001 */  
a {  
    background-color: red;  
}
```

```
/* specificity: 0011 */  
.outer a {  
    background-color: red;  
}
```

```
/* specificity: 0101 */  
#outer a {  
    background-color: red;  
}
```

```
/* specificity: 0111 */  
#outer .inner a {  
    background-color: red;  
}
```

```
/* specificity: 0201 */  
#outer #inner a {  
    background-color: blue;  
}
```

```
/* specificity: 0104 */  
#outer div ul li a {  
    color: yellow;  
}
```

```
/* specificity: 0113 */  
#outer div ul .nav a {  
    color: white;  
}
```

## ¿Y si tiene la misma especificidad?

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

```
/* Esta regla gana sobre la  
anterior */
```

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

El siguiente factor que determina qué regla vence es el orden del código:

Las últimas reglas prevalecen sobre las primeras

!important

```
.mi-clase {  
    color: red!important;  
}
```

# Herencia

Aparte de conocer los selectores y su especificidad, es importante entender la herencia para conocer qué estilos se aplicarán a los elementos.

Existen propiedades CSS que se heredan a los descendientes de forma automática (p.ej. font-family)

# Herencia = Sentido común

¿Qué elementos se heredan? Existe un listado de elementos que se heredan, pero siempre siguen el sentido común.

Por ejemplo, propiedades como “color” o “font-family” se heredan, y otras como “margin”, “padding” o “border” no.

## Propiedades abreviadas

Es posible agrupar algunas propiedades en una única, indicando los valores separados por un espacio:

```
.principal {  
    border: 2px solid #f00;  
}
```



## Margin & Padding

```
#div {  
margin-top: 0;  
margin-right: 5px;  
margin-bottom: 10px;  
margin-left: 15px;  
(auto, 0, px, pt, em or %)  
}
```

```
#div {  
margin:0 5px 10px 15px;  
(top right bottom left)  
}
```

```
#div {  
margin-top: 10px;  
margin-right: 20px;  
margin-bottom: 0;  
margin-left: 20px;  
}
```

```
#div {  
margin:10px 20px 0;  
(top right/left bottom)  
}
```

```
#div {  
margin-top: 0;  
margin-right: auto;  
margin-bottom: 0;  
margin-left: auto;  
}
```

```
#div {  
margin:0 auto;  
(top/bottom left/right)  
}
```

```
#div {  
margin-top: 50px;  
margin-right: 50px;  
margin-bottom: 50px;  
margin-left: 50px;  
}
```

```
#div {  
margin:50px;  
(top/right/bottom/left)  
}
```

## Border

```
#div {  
border-width: 5px;  
(thin, thick, medium or set value) (default = medium)  
border-style: dotted;  
(solid, dashed, dotted, double, etc) (default = none)  
border-color: blue;  
(named, hex, rgb or 0-255) (default = value of elements/  
elements parent color property)  
}
```

```
#div {  
border:5px dotted blue;  
}
```

```
#div {  
border-right-width: 2px;  
border-right-style: solid;  
border-right-color:  
#666666;  
}
```

```
#div {  
border-right:2px solid #666;  
}
```

```
#div {  
border-top-width: 3px;  
border-right-width: 2px;  
border-bottom-width: 3px;  
border-left-width: 2px;  
}
```

```
#div {  
border-width:3px 2px;  
}
```

## Background

```
#div {  
background-color: #CCCCCC;  
(named, hex, rgb or 0-255) (default = transparent)  
background-image: url(images/bg.gif);  
(url or none) (default = none)  
background-repeat: no-repeat;  
(repeat, repeat-x, repeat-y or no-repeat) (default = repeat)  
background-attachment: scroll;  
(fixed or scroll) (default = scroll)  
background-position: top left;  
(top, right, left, bottom or center) (default = 0% 0%)  
}
```

```
#div {  
background: #CCC url(images/bg.gif) no-repeat 0 0;  
}
```

## Font

```
#div {  
font-family: Verdana, Arial, Helvetica;  
(Verdana, Arial, "Times New Roman", etc) (default = browse based)  
font-size: 12px;  
(xx-small, medium, x-large, set value, etc) (default = medium)  
font-weight: bold;  
(normal, bold, bolder, lighter, 100-900 or inherit) (default = normal)  
font-style: italic;  
(normal, italic or oblique) (default = normal)  
font-variant: normal;  
(normal or small-caps) (default = normal)  
line-height: 1.5px;  
(normal, px, pt, em or %) (default = normal)  
}
```

```
#div {  
font: italic bold 12px/1.5px Verdana, Arial, Helvetica;  
}
```

## List

```
#div {  
list-style-image: url(images/bullet.gif);  
(url or none) (default = none)  
list-style-position: inside;  
(inside or outside) (default = outside)  
list-style-type: square;  
(circle, disc, square, etc) (default = disc)  
}
```

```
#div {  
list-style: square inside url(images/bullet.gif);  
}
```

## Color

Aqua: #00ffff to #0ff  
Black: #000000 to #000  
Blue: #0000ff to #00f  
Dark Grey: #666666 to #666  
Fuchsia: #ff00ff to #f0f  
Light Grey: #cccccc to #ccc  
Lime: #00ff00 to #0f0  
Orange: #ff6600 to #f60  
Red: #ff0000 to #f00  
White: #ffffff to #fff  
Yellow: #ffff00 to #ff0

# Display

Los elementos tienen un tipo de caja por defecto que afecta a su presentación, por ejemplo block o inline.

La propiedad display permite alterar el tipo de representación del elemento

```
div {  
    display: inline;  
    display: inline-block;  
    display: block;  
    display: none;  
}
```

# display: inline

- Es el valor por defecto de la mayoría de elementos.
- Acepta margin y padding, pero no empuja a otros elementos verticalmente.
- No acepta las propiedades height y width.



Pellentesque *inline element* morbi tristique senectus  
Donec eu libero sit amet quam egestas semper. Aenean

Pellentesque *inline element* morbi tristique senectus et netus et  
malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae,  
ultrices eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas  
semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.

# display: inline-block

- Al igual que inline, se posiciona de forma natural en la misma línea.
- La mayor diferencia es que adquiere otras propiedades de block, como permitir especificar height y width



## display: block

- Algunos elementos son block por defecto (div, section, ul,...)
- Normalmente son contenedores, aunque también pueden ser de texto (p, h1,...)
- Se sitúan en una nueva línea y por defecto adquieren todo el ancho disponible.



# Posicionamiento

La propiedad CSS `position` nos permite alterar la posición natural de los elementos en la página.

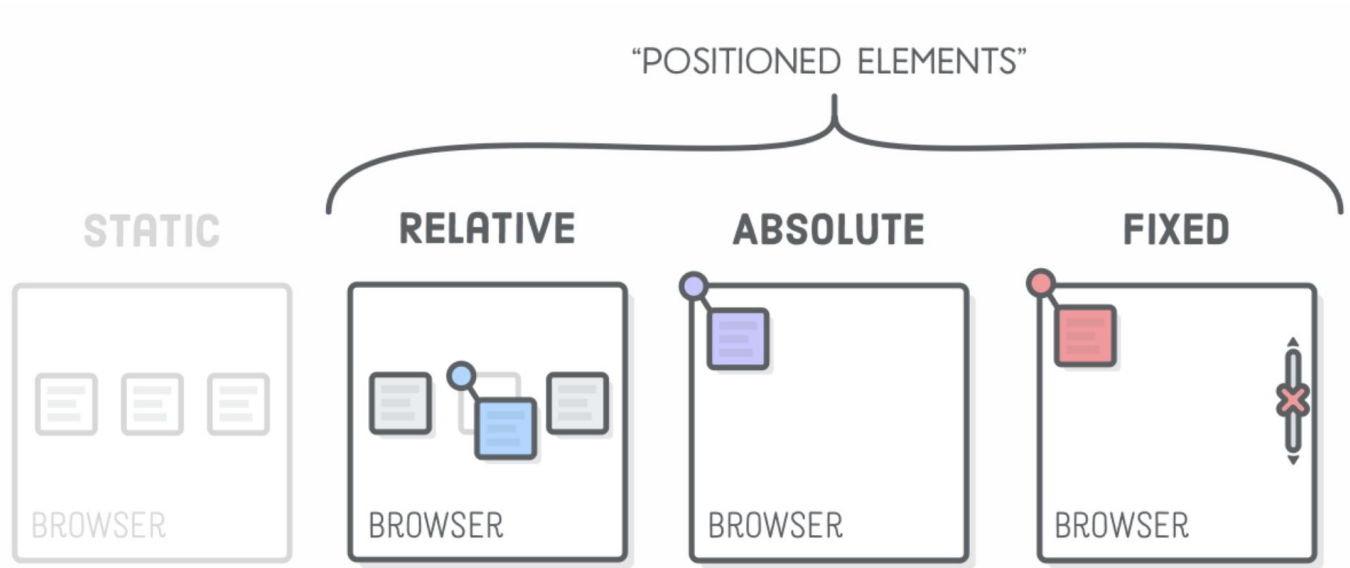
El valor por defecto es `static`, y los elementos que tengan otro valor los llamaremos “elementos posicionados”.

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

```
div {  
    position: relative;  
}
```

```
div {  
    position: absolute;  
}
```

# Posicionamiento

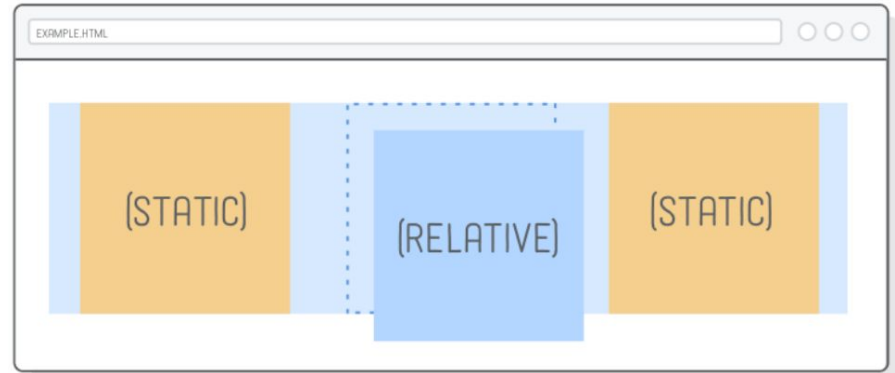


Fuente: <https://internetingishard.com/html-and-css/advanced-positioning/#positioned-elements>



# Posición relativa

- La "posición relativa" mueve elementos alrededor de donde normalmente aparecerían en el flujo estático de la página.



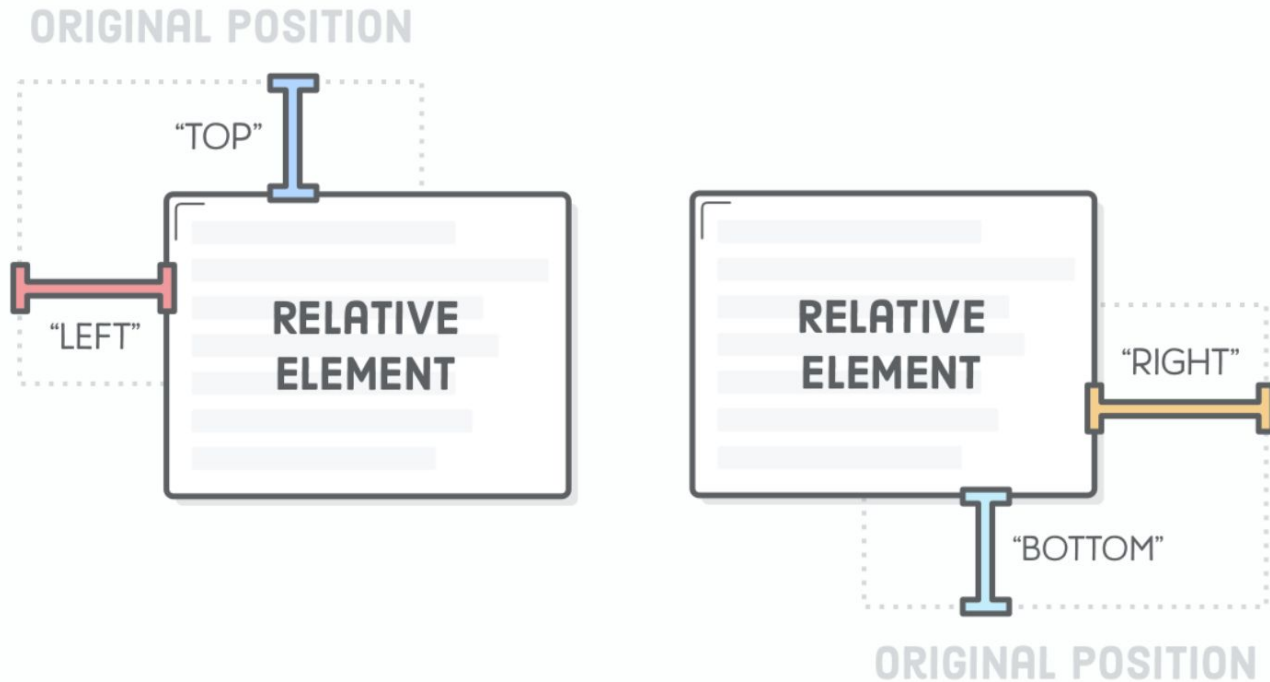
# Posición relativa

- Utilizamos las siguientes propiedades para indicar las coordenadas (el *offset*) del elemento respecto al origen:
  - top
  - right
  - bottom
  - left

```
em {  
    position: relative;  
    top: 2em;  
    left: 2em;  
}
```

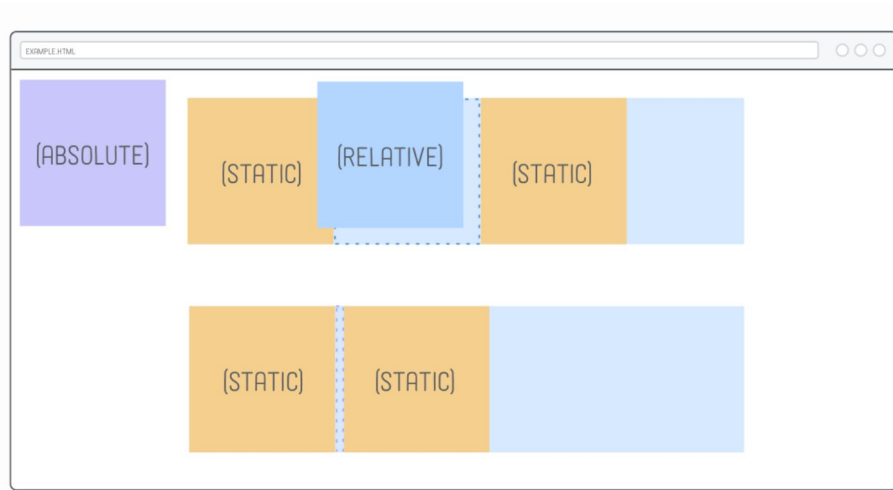
```
em {  
    position: relative;  
    bottom: 100px;  
    right: 20px;  
}
```

# Posición relativa



# Posición absoluta

- La "posición absoluta" sale completamente del flujo normal y se puede posicionar en cualquier lugar de la página.



# Posición absoluta

- Utilizamos las siguientes propiedades para indicar las coordenadas del elemento respecto al origen:
  - top
  - right
  - bottom
  - left

```
em {  
    position: absolute;  
    top: 2em;  
    right: 100px;  
}
```

# Posición (relativamente) absoluta

- Cuando el elemento absoluto es hijo de un elemento relativo, se posiciona respecto al padre.
- Las coordenadas de un elemento absoluto se calculan respecto al elemento posicionado (no estático) más cercano.



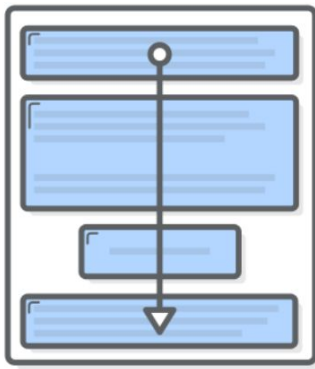
# Fixed

- Es muy similar al absoluto: también se sale del flujo natural de la página.
- Su posicionamiento siempre es respecto al navegador.
- Aunque hagamos scroll, se queda fijo en el mismo sitio.

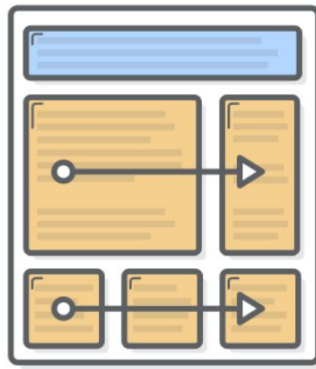
```
div {  
    position: fixed;  
    top: 0;  
    left: 100px;  
}
```

# Float

Hasta ahora los elementos de tipo bloque seguían el flujo vertical de la página. La propiedad CSS `float` permite posicionar elementos de tipo bloque unos junto a otros.



**VERTICAL FLOW**



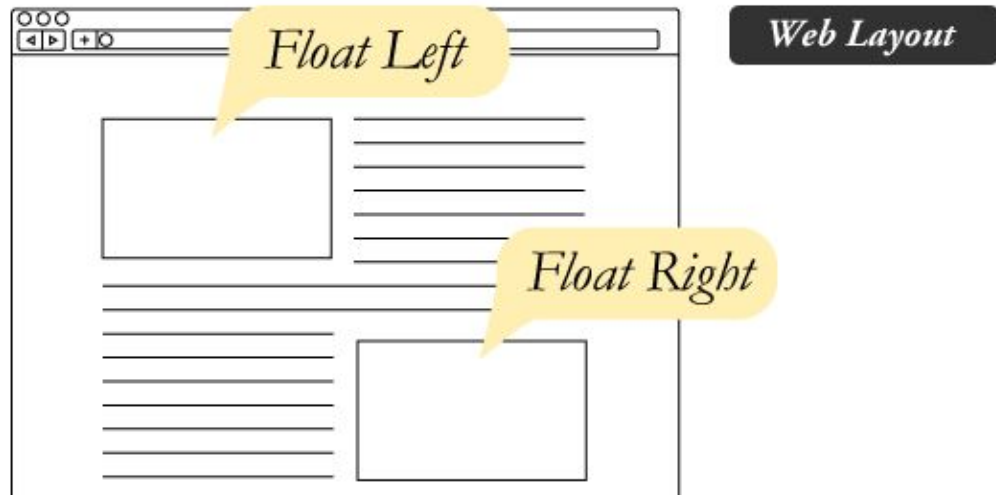
**HORIZONTAL FLOW**



# Float

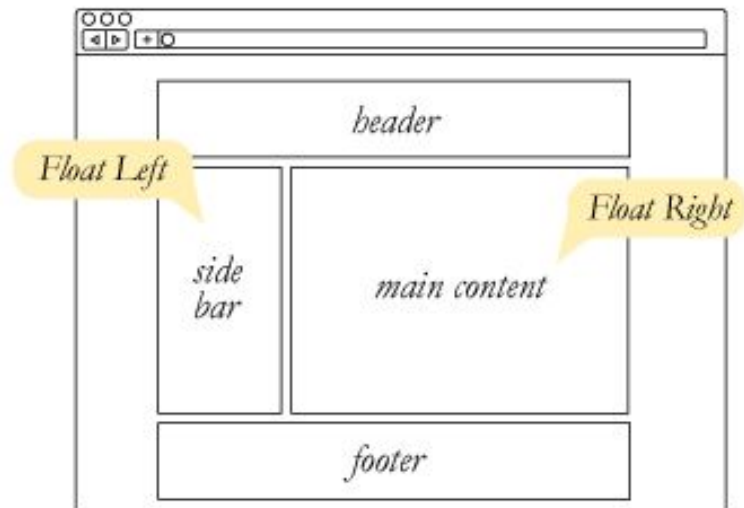
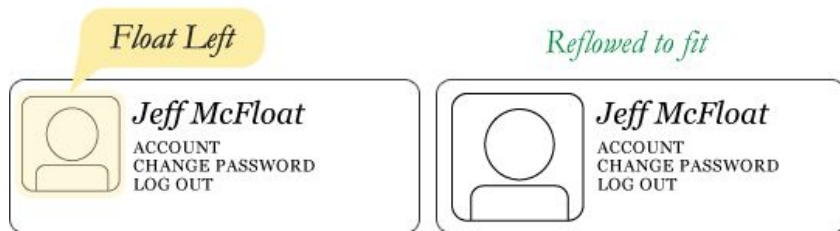
Hay 4 valores distintos:

- **left y right** para llevar los elementos en esas direcciones.
- **none**: el valor por defecto (no altera el comportamiento)
- **inherit**: hereda el mismo valor que el padre.



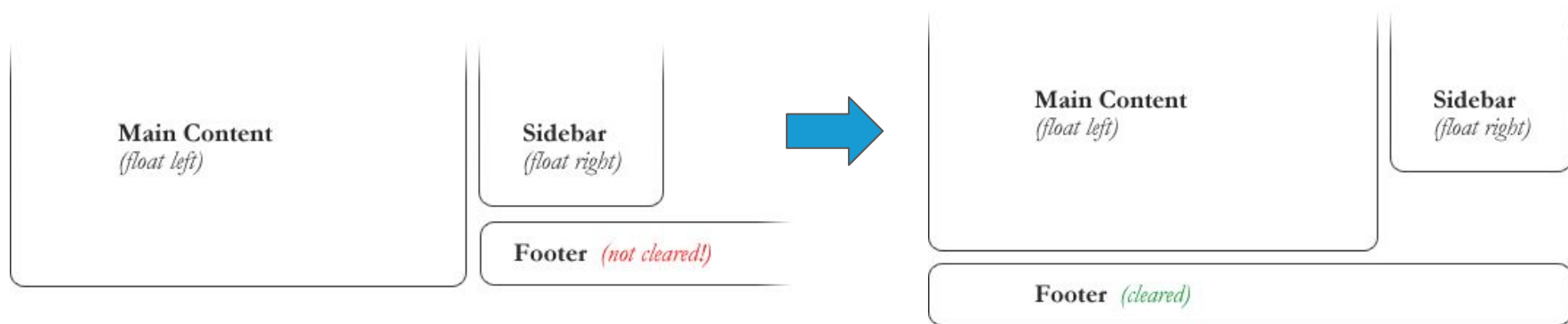
# ¿Cuál es su uso?

- Se pueden utilizar para construir layouts completos o pequeñas estructuras.

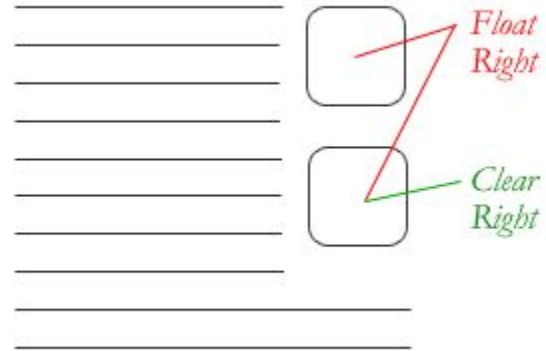
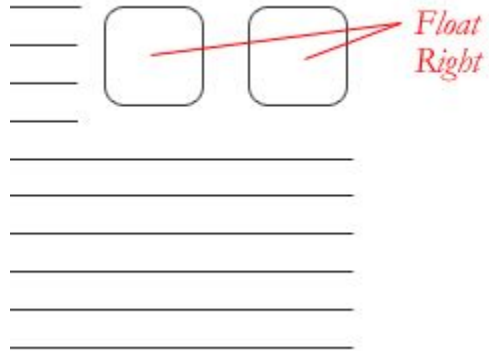


## El anti-float: clear

Indica qué lados de la caja no deben ser adyacentes a un elemento posicionado con float. Sus posibles valores son: **left**, **right**, **both** y **none**

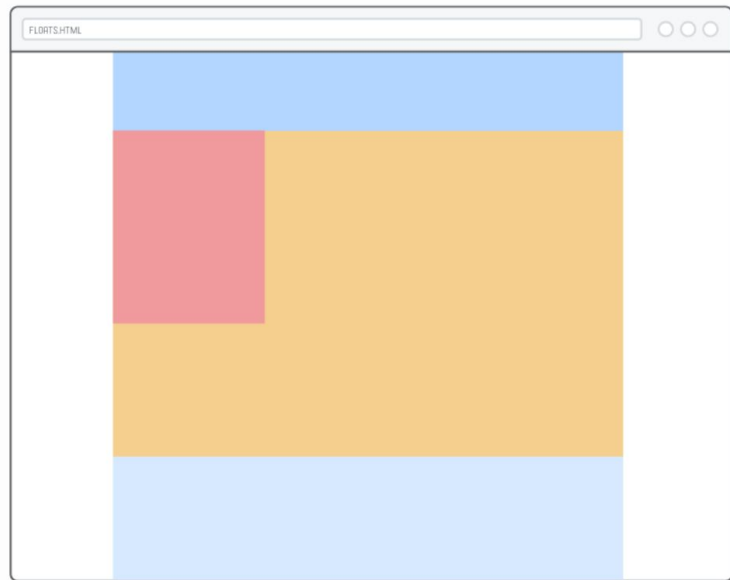


# El anti-float: clear

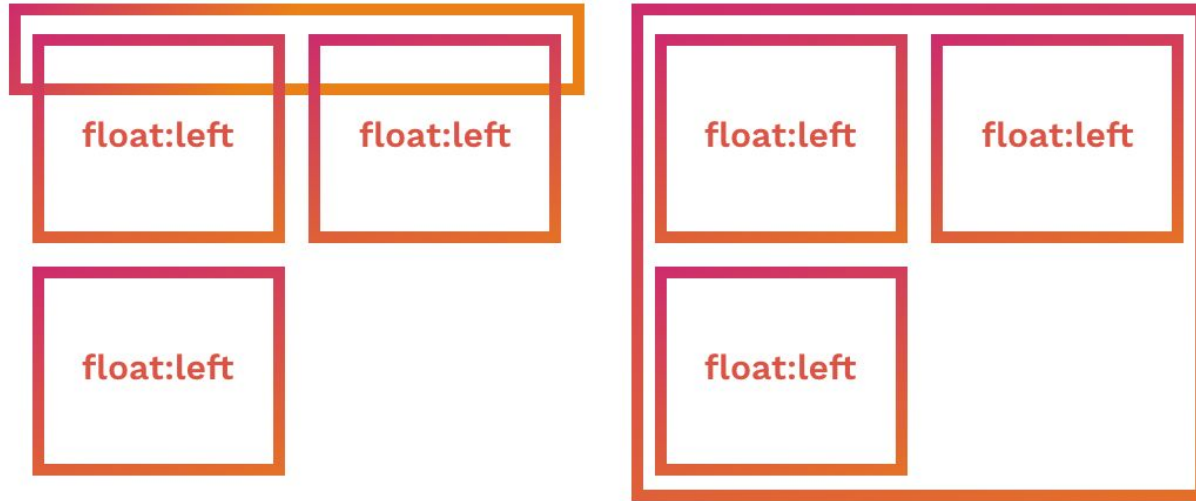


# Floats en página centrada

- Normalmente tendremos un ancho fijo de página y lo centraremos (margin: 0 auto).
- Luego utilizaremos “float” para crear estructuras de 2 o 3 columnas.
- Finalmente utilizaremos “clear” para posicionar el footer.



# Forzar clear en el contenedor



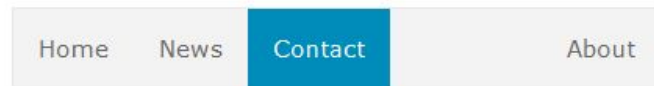
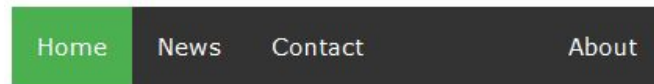
## Forzar clear en el contenedor

- En ocasiones es necesario forzar el clear después de los hijos del contenedor.
- Se utiliza la pseudoclase :after, el cual permite añadir contenido al final del contenedor.

```
.container:after {  
    content: ".";  
    visibility: hidden;  
    display: block;  
    height: 0;  
    clear: both;  
}
```

# Barras de menú

- Aunque no lo parezca, se utilizan listas en las que cada elemento es un hipervínculo.
- Para ello se utiliza la propiedad: `list-style-type: none;`
- Su apariencia puede ser horizontal o vertical .





```
ul {  
    list-style-type: none;  
    background-color: #333;  
}
```

```
li {  
    display: inline-block;  
}
```

```
li a {  
    display: block;  
    color: white;  
    text-align: center;  
    padding: 14px 16px;  
    text-decoration: none;  
}
```

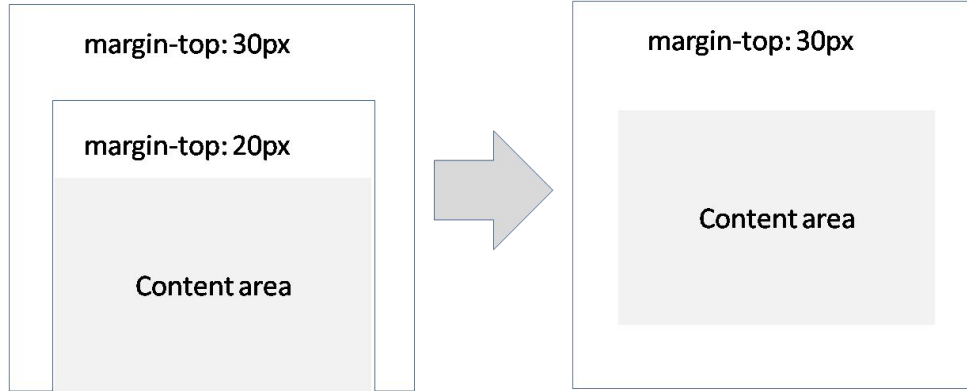
```
<ul>  
    <li><a href="#home">Home</a></li>  
    <li><a href="#news">News</a></li>  
    <li><a href="#cont">Contact</a></li>  
    <li><a href="#about">About</a></li>  
</ul>
```

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    width: 200px;  
    background-color: #f1f1f1;  
}
```

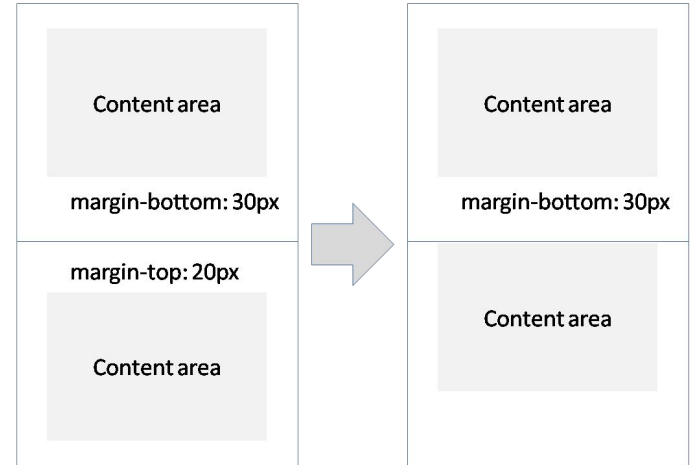
```
li a {  
    display: block;  
    color: #000;  
    padding: 8px 16px;  
    text-decoration: none;  
}
```

```
<ul>  
    <li><a href="#h">Home</a></li>  
    <li><a href="#n">News</a></li>  
    <li><a href="#c">Contact</a></li>  
    <li><a href="#a">About</a></li>  
</ul>
```

# Problemas con márgenes: Collapsing margins



## Margin collapsing



## Problemas con márgenes: Collapsing margins

- Los márgenes Top y Bottom de los bloques a veces se “*colapsan*” en un solo margen, cuyo tamaño es el mayor de los márgenes
- Los márgenes de flotantes y elementos con posición absoluta nunca colapsan.
- Cuando los elementos tienen borde o padding, no se solapan.
- Una opción para solucionarlo es hacer que el elemento sea “float” y ocupe el 100% (width) o convertirlo a tipo inline-block

# Sources

- Mozilla MDN Web Docs: <https://developer.mozilla.org/>
- HTML Dog: <http://htmldog.com>
- Interneting is Hard: <https://internetingishard.com/>