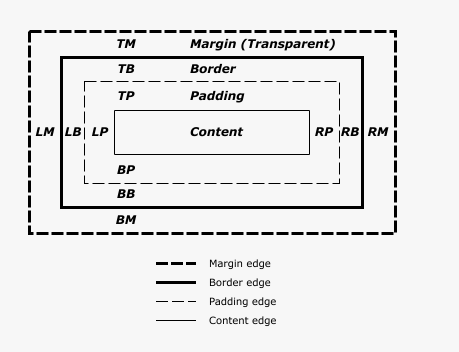
**MEMORIA CSS**

Las propiedades y sus valores forman bloques declarativos, y cómo los bloques declarativos y los selectores forman reglas CSS completas

<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>



BOX MODEL



* **Selectores simples**: Seleccionan los elementos por el nombre del tipo de elemento, [class](https://developer.mozilla.org/es/docs/Web/HTML/Global_attributes" \l "attr-class), o su [id](https://developer.mozilla.org/es/docs/Web/HTML/Global_attributes#attr-id).
* **Selectores de atributos**: Seleccionan los elementos por los valores de sus atributos.
* **Pseudo-clases**: Seleccionan los elementos por el estado en que se encuentran, cómo haber aparecido al pasar el ratón, o el tic deshabilitado o seleccionado, o por ser el primer hijo de su padre en el árbol DOM.
* **Pseudo-elementos**: Selecciona los elementos por su situación en relación a otro elemento, por ejemplo: la primera palabra de cada párrafo, o el contenido que se encuentra justo después de un elemento.
* **Combinaciones**: No son en sí mismos selectores, sino formas de combinar dos o más selectores de forma práctica para una selección especial. Por ejemplo, se pueden seleccionar párrafos que sean descendientes de divs, o párrafos situados justo después de títulos.

**Selectores múltiples**: Tampoco son selectores en sí mismos; podemos agrupar múltiples selectores en la misma regla CSS separados por comas, para aplicarlos a una de las declaraciones o a todos los elementos seleccionados por estos selectores.

Selector de atributo:



Pseudo elementos : <https://developer.mozilla.org/es/docs/Web/CSS/Pseudoelementos>

Estos selectores de atributos afectarán a los elementos cuyo valor coincida exactamente con el valor del atributo especificado:

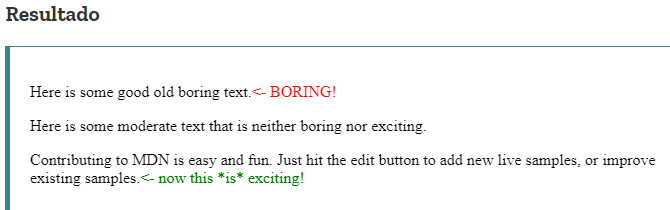
* **[attr]** : Este selector ‘seleccionará’ todos los elementos que contengan el atributo attr, sin importar el valor que tenga.
* **[attr=val]** : Este, seleccionará los elementos con el atributo attr, pero solo aquello cuyo valor coincida con val.
* **[attr~=val]**: Este selector afectará a los elementos con el atributo attr, pero solo si el valor val está contenido en la lista de valores (separados por espacios) incluidos en el valor de attr, por ejemplo una de las clases contenida en una lista de clases (separadas por espacios).

Al igual que en otros lenguajes podemos realizar reglas específicas para ciertos elementos aplicando “Expresiones Regulares”.

De ahí los selectores conocidos como «***RegExp-like selectors***«, pues proporcionan una forma de selección similar a las expresiones regulares (aunque siendo estrictos, estos selectores no son verdaderas expresiones regulares):

**[attr|=val]** : Este selector elegirá todos los elementos con el atributo attr cuyo valor sea exactamente val o empieza por val- (nota: el guion no es un error, se usa para manejar códigos de lenguaje de programación).  
  
**[attr^=val]** : Seleccionará todos los elementos cuyo atributo attr **comienza** por el valor val.  
**[attr$=val]** : Este selector elegirá todos los elementos cuyo atributo attr **termina** por el valor val.  
**[attr\*=val]** : Este seleccionará todos los elementos cuyo atributo attr **contiene** la cadena val (al contrario que [attr~=val], este selector no considera los espacios como separador de valores sino como parte del valor del atributo).





La **Especificidad** es en sí una medida de cómo de específico es un selector — cuántos elementos puede seleccionar. Como hemos mostrado en el ejemplo anterior, los selectores de elementos son poco específicos. Los selectores de clase son más específicos, por lo que prevalecen sobre los selectores de elementos. Los selectores ID son todavía más específicos, por lo tanto ganan frente a los de clase. La única forma de vencer a los selectores ID es usando la declaración !important.

La especificidad que tiene un selector se mide mediante 4 valores (o componentes) diferentes, podemos pensar en ellos como en 4 columnas de unidades de millar, centenas, decenas y unidades:

1. **Unidades de millar:** Puntúa 1 en esta columna si la declaración está dentro de un atributo [style](https://developer.mozilla.org/es/docs/Web/HTML/Global_attributes" \l "attr-style) (como las declaraciones que no tienen selectores, que su especificidad es siempre 1000). Sino puntúa 0.
2. **Centenas:** Puntúa 1 en esta columna por cada selector ID contenido en el selector.
3. **Decenas:** Puntúa 1 en esta columna para cada selector de clase, selector de atributo o de pseudo-clase contenidos en el selector.
4. **Unidades:** Puntúa 1 en esta columna por cada selector de elemento o pseudo-elemento contenidos en el selector.

En CSS, podemos combinar varios selectores juntos y con ello seleccionar elementos contenidos en otros elementos, o elementos adyacentes a otros. Disponemos de cuatro tipos:

* **El selector descendiente** —  (espacio) — permite seleccionar un elemento anidado en alguna parte dentro de otro elemento (no tiene por qué ser un hijo; puede ser un nieto, por ejemplo)
* **El selector hijo** — > — permite seleccionar un elemento que es hijo directo de otro elemento.
* **El selector hermano** — + — permite seleccionar un elemento que es hermano directo de otro elemento (a la derecha por ejemplo, en el mismo nivel jerárquico).
* **El selector hermano en general** — ~ — permite seleccionar cualquier elemento hermano de otro (por ejemplo en el mismo nivel jerárquico, pero no necesariamente adyacente a él).

<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>

### **Control de la herencia**

CSS dispone de tres valores especiales para manejar las herencias:

* [inherit](https://developer.mozilla.org/es/docs/Web/CSS/inherit) : Este valor establece el valor de la propiedad de un elemento seleccionado en el mismo que su elemento padre.
* [initial](https://developer.mozilla.org/es/docs/Web/CSS/initial) : Este valor establece el valor de la propiedad de un elemento seleccionado en el valor por defecto que establece la hoja de estilos del navegador, si este no existe, la propiedad se hereda naturalmente, adoptando el valor de inherit.
* [unset](https://developer.mozilla.org/es/docs/Web/CSS/unset) : Este valor reestablece la propiedad a su valor natural, esto es: si la propiedad se hereda de forma natural entonces actuará como inherit, sino, actuará como initial.

El valor inherit es el más interesante — nos permite, de forma explícita, hacer que un elemento herede de su padre el valor de una propiedad.

## Por defecto, un elemento de bloque está al 100% de la anchura de su elemento padre y tan alto como su contenido.

## Los elementos en línea son tan altos y anchos como su contenido.

A estos últimos elementos no puedes definirles su altura o su anchura. Como ya te adelanté en su momento, para realizar esto había “trucos”. Básicamente hay que aplicarles una regla css que indique que se comporte como elementos de bloque, aunque realmente sea un elemento inline (**display**: *inline-block*o **display**: *block*).

¿Qué pasa con los elementos que tienen interacción con otros?. Si atendemos al flujo normal, unos quedarían encima de otros, o terriblemente descolocados, etc.

Cuando utilizas CSS para crear un diseño, lo que está haciendo es alejar los elementos de este flujo normal, sin embargo, para muchos de los elementos en su página, este flujo normal creará exactamente el diseño que necesita. Esta es la razón por la cual comenzar con un documento HTML bien estructurado es tan importante, ya que puede trabajar con la forma en que las cosas se presentan por defecto en lugar de luchar contra ella.

Los métodos que pueden cambiar cómo se organizan los elementos en CSS son los siguientes:

* **La propiedad**[**display**](https://developer.mozilla.org/es/docs/Web/CSS/display) — Valores estandar como block, inline or inline-block puede cambiar cómo se comportan los elementos en flujo normal (ver [Types of CSS boxes](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Box_model" \l "Types_of_CSS_boxes) para mas informacion).
* **Floats** — Aplicando un valor [float](https://developer.mozilla.org/es/docs/Web/CSS/float) como left puede hacer que los elementos de nivel de bloque se envuelvan a un lado de un elemento, como la forma en que las imágenes a veces tienen texto flotando a su alrededor en diseños de revistas.
* **La propiedad**[**position**](https://developer.mozilla.org/es/docs/Web/CSS/position) — Le permite controlar con precisión la ubicación de las cajas dentro de otras cajas. El posicionamiento static es por defecto en un flujo normal, pero puede hacer que los elementos se distribuyan de forma diferente utilizando otros valores, por ejemplo, siempre fijados en la parte superior izquierda de la ventana del navegador.
* **Table layout** — las características diseñadas para diseñar las partes de una tabla HTML se pueden usar en elementos que no son de tabla usando display: table y propiedades asociadas.
* **Multi-column layout** — La propiedad [Multi-column layout](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Columns) puede hacer que el contenido de un bloque se distribuya en columnas, como puede ver en un periódico.

