



# Preparador Informática

[www.preparadorinformatica.com](http://www.preparadorinformatica.com)

## MANUAL 2 PROGRAMACIÓN WEB

### CSS

|  |           |
|--|-----------|
| <b>1. Introducción</b>                           | <b>2</b>  |
| <b>1.1. Formas de utilizar CSS</b>               | <b>3</b>  |
| <b>1.2. Estructura de CSS</b>                    | <b>4</b>  |
| <b>1.3. Versiones de CSS</b>                     | <b>5</b>  |
| <b>1.4. Herencia en CSS</b>                      | <b>6</b>  |
| <b>2. Propiedades CSS</b>                        | <b>7</b>  |
| <b>2.1. Colores</b>                              | <b>7</b>  |
| <b>2.2. Unidades</b>                             | <b>12</b> |
| <b>2.3. Modelo de cajas</b>                      | <b>14</b> |
| <b>2.4. Bordes</b>                               | <b>17</b> |
| <b>2.5. Fondos</b>                               | <b>21</b> |
| <b>2.6. Tipos de elementos</b>                   | <b>23</b> |
| <b>2.7. Fuentes y tipografías</b>                | <b>24</b> |
| <b>2.8. Alineaciones y variaciones del texto</b> | <b>27</b> |
| <b>2.9. Tablas CSS</b>                           | <b>28</b> |
| <b>2.10. Listas CSS</b>                          | <b>28</b> |
| <b>3. Selectores CSS</b>                         | <b>30</b> |
| <b>3.1. Selectores CSS básicos</b>               | <b>30</b> |
| <b>3.2. Selectores CSS avanzados</b>             | <b>33</b> |
| <b>3.3. Pseudoclasas CSS</b>                     | <b>39</b> |
| <b>3.4. Pseudoclasas CSS avanzadas</b>           | <b>41</b> |
| <b>3.5. Pseudoelementos CSS</b>                  | <b>44</b> |
| <b>3.6 Atributos CSS</b>                         | <b>46</b> |
| <b>4. Cascada en CSS</b>                         | <b>47</b> |
| <b>5. “Chuleta” de CSS</b>                       | <b>49</b> |
| <b>6. Sololearn</b>                              | <b>49</b> |
| <b>7. Codepen</b>                                | <b>50</b> |
| <b>8. W3schools</b>                              | <b>50</b> |
| <b>9. Bibliografía</b>                           | <b>50</b> |



## 1. Introducción

Las siglas **CSS** (*Cascading Style Sheets*) significan «Hojas de estilo en cascada» y parten de un concepto simple: aplicar **estilos** (colores, formas, márgenes, etc...) a uno o varios documentos HTML de forma masiva.



Con HTML y CSS se busca la separación de presentación y contenido. De esta forma, se puede unificar todo lo relativo al diseño visual en **un solo documento CSS**, y con ello, se tienen varias ventajas:

- Si necesitamos hacer modificaciones visuales lo hacemos en un sólo lugar y no tenemos que editar todos los documentos HTML en cuestión por separado.
- Se reduce la duplicación de estilos en diferentes lugares, por lo que es más fácil de organizar y hacer cambios. Además, al final la información a transmitir es considerablemente menor (las páginas se descargan más rápido).
- Es más fácil crear versiones diferentes de presentación para otros tipos de dispositivos: tablets, smartphones o dispositivos móviles, etc...

## 1.1. Formas de utilizar CSS

Existen tres formas para incluir contenido CSS en un documento HTML:

### *CSS externo*

Se incluye en la cabecera del documento HTML una relación al archivo CSS que contiene los estilos. Se hace a través de la etiqueta <link>

En la cabecera del HTML (<head></head>) se incluye una relación al archivo CSS en cuestión:

Ejemplo:

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

### *CSS interno*

Se añaden los estilos CSS directamente en la cabecera HTML del documento entre las etiquetas <style>

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Título de la página</title>
  <style type="text/css">
    div {
      background:#FFFFFF;
    }
  </style>
</head>
...
```

### *CSS embebido*

Consiste en aplicar estilos directamente en las propias etiquetas utilizando el atributo **style**:

Ejemplo:

```
<p>¡Hola <span style="color:#FF0000">mundo</span>!</p>
```



## 1.2. Estructura de CSS

La **estructura CSS** se basa en reglas que tienen el siguiente formato:



- **Selector:** El selector es el elemento HTML que vamos a seleccionar del documento para aplicarle un estilo concreto. *Por ejemplo, puede ser la etiqueta <p>.*
- **Propiedad:** La propiedad es una de las diferentes características que brinda el lenguaje CSS para actuar sobre el selector.
- **Valor:** Cada propiedad CSS tiene una serie de valores concretos, con los que tendrá uno u otro comportamiento.

Ejemplo:

Código HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Título de página</title>
    <link rel="stylesheet" type="text/css" href="index.css" />
  </head>
  <body>
    <div id="first">
      <p>Párrafo</p>
    </div>
    <div id="second">
      <span>Capa</span>
    </div>
  </body>
</html>
```

Código CSS

```
p {
  color: red;      /* Color de texto rojo */
}
```

El navegador al cargar el código HTML y CSS anterior mostrará:

Párrafo

Capa



Cada selector no tiene que tener una sola regla, puede tener muchas reglas asociadas. El último ; de un selector (en naranja) no es obligatorio y se puede omitir



Para hacer más legible el código CSS, se suele utilizar la siguiente estructura visual:

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

### 1.3. Versiones de CSS

El **lenguaje CSS** es una especificación desarrollada y mantenida por el World Wide Web Consortium (W3C), una comunidad internacional que se encarga de desarrollar estándares para asegurar el crecimiento y la neutralidad de la web, independizándolo de tecnologías propietarias e intentando aunar esfuerzos para satisfacer la demanda de características útiles e interesantes.

En el consorcio participan y colaboran prácticamente casi todas las empresas relacionadas con Internet, como por ejemplo Apple, Adobe, Akamai, Cisco, Google, Facebook, HP, Intel, LG, Microsoft, Nokia, Twitter, Yahoo, entre muchos otros.

A lo largo de su historia, **CSS** ha evolucionado en diferentes versiones, denominados **niveles**:

| Nivel         | Año  | Descripción   |
|---------------|------|---|
| <b>CSS1</b>   | 1996 | Propiedades de fuente, colores, alineación, etc...      |
| <b>CSS2</b>   | 1998 | Propiedades de posicionamiento, tipos de medios, etc... |
| <b>CSS2.1</b> | 2005 | Corrige errores de CSS2 y modifica ciertas propiedades  |
| <b>CSS3</b>   | 2011 | Inicio de módulos separados con funcionalidades nuevas  |

## 1.4. Herencia en CSS

Las propiedades CSS se aplican desde arriba hacia abajo, sobrescribiendo las propiedades anteriores que se repitan.

En primer lugar, debemos saber que algunas propiedades CSS se **heredan** desde los elementos padres a los elementos hijos, modificando el valor que tienen por defecto:

Ejemplo: Por defecto la propiedad **color** hereda de los padres a los hijos

```
body {  
  color: green;    /* Color de texto verde */  
}
```

La herencia no ocurre con todas las propiedades CSS, sino sólo con algunas propiedades como **color** o **font**, donde sí suele ser deseable. Hay otras propiedades CSS que no heredan, como **los bordes** de un elemento HTML:

Ejemplo: Por defecto la propiedad **border** no hereda de los padres a los hijos

```
body {  
  border-width: 2px;  
  border-style: solid;  
  border-color: red;  
}
```

### Valores especiales

| Valor          | Significado   |
|----------------|---|
| <b>inherit</b> | Hereda el valor de la propiedad del elemento padre.   |
| <b>initial</b> | Establece el valor que tenía la propiedad inicialmente.   |
| <b>unset</b>   | Combinación de las dos anteriores: Hereda el valor de la propiedad del elemento padre, y en caso de no existir, su valor inicial. |

Ejemplo: Uso de inherit para forzar la herencia:

```
body {  
  border-width: 2px;  
  border-style: solid;  
  border-color: red;  
}  
  
p {  
  border: inherit;  
}
```

Si tenemos un elemento **<p>** dentro del **<body>**, el primero heredará los estilos del elemento **<body>**, ya que le hemos especificado el valor **inherit** en la propiedad **border**.



## 2. Propiedades CSS

### 2.1. Colores

Para hacer variaciones en los colores del texto y del fondo se utilizan las siguientes propiedades:

| Propiedad         | Valor        | Significado  |
|-------------------|--------------|--|
| color:            | <b>COLOR</b> | Cambia el <b>color del texto</b> que está en el interior de un elemento. |
| background-color: | <b>COLOR</b> | Cambia el <b>color de fondo</b> de un elemento.                          |

#### Colores CSS

La propiedad **color** establece el color del texto, mientras que **background-color** establece el color de fondo del elemento.

Todas las propiedades CSS donde existen valores **COLOR**, establecen la posibilidad de indicar **4 formas alternativas** para especificar el color deseado:

| Nombre                                 | Formato  | Ejemplo                   |
|--|--|---------------------------|
| Palabra clave predefinida              | <i>[palabra]</i>   | red                       |
| Esquema RGB                            | rgb( <i>rojo</i> , <i>verde</i> , <i>azul</i> )                        | rgb(255, 0, 0)            |
| Esquema RGB con canal alfa             | rgba( <i>rojo</i> , <i>verde</i> , <i>azul</i> , <i>alfa</i> )         | rgba(255, 0, 0, 0.25)     |
| Esquema RGB hexadecimal                | # <i>RRGGBB</i>  | #ff0000                   |
| Esquema RGB hexadecimal con canal alfa | # <i>RRBBBAA</i>   | #ff000040                 |
| Esquema HSL                            | hsl( <i>color</i> , <i>saturación</i> , <i>brillo</i> )                | hsl(0, 100%, 100%)        |
| Esquema HSL con canal alfa             | hsla( <i>color</i> , <i>saturación</i> , <i>brillo</i> , <i>alfa</i> ) | hsla(0, 100%, 100%, 0.25) |

#### Palabras clave de color

El primer caso (y más limitado) permite establecer el color utilizando palabras reservadas de colores, como **red**, **blue**, **orange**, **white**, **navy**, **yellow** u otras.

Existen más de 140 palabras clave para indicar colores:

(ver en la siguiente página)





|                |                 |                  |                      |
|----------------|-----------------|------------------|----------------------|
| black          | navy            | darkblue         | mediumblue           |
| blue           | darkgreen       | green            | teal                 |
| darkcyan       | deepskyblue     | darkturquoise    | mediumspringgreen    |
| lime           | springgreen     | aqua             | cyan                 |
| midnightblue   | dodgerblue      | lightseagreen    | forestgreen          |
| seagreen       | darkslategray   | darkslategrey    | limegreen            |
| mediumseagreen | turquoise       | royalblue        | steelblue            |
| darkslateblue  | mediumturquoise | indigo           | darkolivegreen       |
| cadetblue      | cornflowerblue  | mediumaquamarine | dimgray              |
| dimgray        | slateblue       | olivedrab        | slategray            |
| slategrey      | lightslategray  | lightslategrey   | mediumslateblue      |
| lawngreen      | chartreuse      | aquamarine       | maroon               |
| purple         | olive           | gray             | grey                 |
| skyblue        | lightskyblue    | blueviolet       | darkred              |
| darkmagenta    | saddlebrown     | darkseagreen     | lightgreen           |
| mediumpurple   | darkviolet      | rebeccapurple    | palegreen            |
| darkorchid     | yellowgreen     | sienna           | brown                |
| darkgray       | darkgrey        | lightblue        | greenyellow          |
| paleturquoise  | lightsteelblue  | powderblue       | firebrick            |
| darkgoldenrod  | mediumorchid    | rosybrown        | darkkhaki            |
| silver         | mediumvioletred | indianred        | peru                 |
| chocolate      | tan             | lightgray        | lightgrey            |
| thistle        | orchid          | goldenrod        | palevioletred        |
| crimson        | gainsboro       | plum             | burlywood            |
| lightcyan      | lavender        | darksalmon       | violet               |
| palegoldenrod  | lightcoral      | khaki            | aliceblue            |
| honeydew       | azure           | sandybrown       | wheat                |
| beige          | whitesmoke      | mintcream        | ghostwhite           |
| salmon         | antiquewhite    | linen            | lightgoldenrodyellow |
| oldlace        | red             | fuchsia          | magenta              |
| deeppink       | orangered       | tomato           | hotpink              |
| coral          | darkorange      | lightsalmon      | orange               |
| lightpink      | pink            | gold             | peachpuff            |
| navajowhite    | moccasin        | bisque           | mistyrose            |
| blanchedalmond | papayawhip      | lavenderblush    | seashell             |
| cornsilk       | lemonchiffon    | floralwhite      | snow                 |
| yellow         | lightyellow     | ivory            | white                |


### Formato RGB

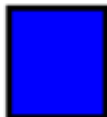
Uno de los métodos más conocidos por los diseñadores gráficos es utilizar el **formato RGB**. Las siglas RGB significan **rojo**, **verde** y **azul**, por lo que cada cifra (*del 0 al 255*) representa la intensidad de cada componente de color. Como se puede ver en la siguiente imagen, si utilizamos una cantidad (0, 0, 0) de cada canal, obtenemos el **color negro**. En cambio, si utilizamos una cantidad (255, 0, 0), obtendremos el **color rojo**.

|              |   |     |     |     |
|--------------|---|-----|-----|-----|
| <b>Rojo</b>  | 0 | 255 | 255 | 0   |
| <b>Verde</b> | 0 | 255 | 0   | 0   |
| <b>Azul</b>  | 0 | 255 | 0   | 255 |

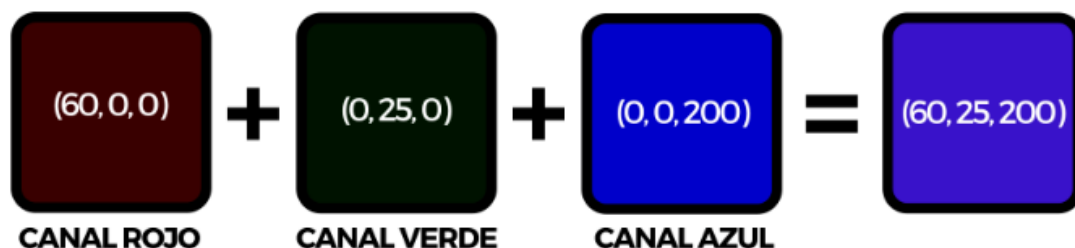
||  


||  


||  


||  


De esta forma, mezclando las cantidades de cada canal, se puede obtener prácticamente cualquier color.



Ejemplo:

```
div {
  background-color: rgb(125, 80, 10);
  color: rgb(255, 255, 0);
}
```

### Formato hexadecimal

El **formato hexadecimal** es el más utilizado por los desarrolladores web, aunque en principio puede parecer algo extraño y complicado, sobre todo si no has oído hablar nunca del sistema hexadecimal (*sistema en base 16 en lugar del que utilizamos normalmente, en base 10*).

Cada par de letras simboliza el valor del RGB en el sistema de numeración hexadecimal, así pues, el color **#FF0000**, o sea HEX(FF,00,00), es equivalente al RGB(255,0,0), que es también equivalente al HSL(0, 100%, 100%). Veamos algunos ejemplos para clarificarlo:

| Hexadecimal | Hex. abreviado | Color RGB   | Palabra clave       |
|-------------|----------------|-------------|---------------------|
| #FF0000     | #F00           | 255,0,0     | red (rojo)          |
| #000000     | #000           | 0,0,0       | black (negro)       |
| #00FFFF     | #0FF           | 0, 255, 255 | cyan (azul claro)   |
| #9370DB     | #97D           | 147,112,219 | mediumpurple (lila) |

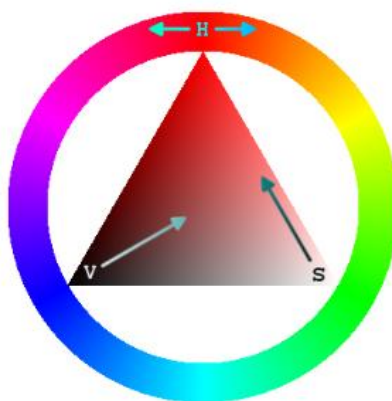
**Truco:** Como se puede ver en la segunda columna, para ahorrar espacio puedes utilizar el **formato hexadecimal abreviado**, especificando sólo las primeras tres cifras de cada par. Por ejemplo, **#9933AA** como **#93A**. El color abreviado sólo será fiel cuando los pares de cifras sean idénticos

#### Ejemplo:

```
div {
  background-color: #512592;
  background-color: #000000; /* Negro (esquema Hex) */
  background-color: #000;    /* Negro (esquema Hex abreviado) */
  background-color: #451;    /* Equivalente a #445511; */
}
```

### Formato HSL

Las siglas HSL significan **color** (o *matiz*), **saturación** y **brillo**. La primera cifra selecciona el matiz de color (*una cifra de 0 a 360 grados*), seleccionando el color del círculo exterior de la imagen. Por su parte, las dos siguientes, son el porcentaje de saturación y el brillo del color, respectivamente (*ambos, porcentajes de 0% a 100%*).



Ejemplo:

```
div {  
  background-color: hsl(35deg, 0%, 100%);  
  background-color: hsl(120deg, 25%, 75%);  
  background-color: hsl(5deg, 20%, 20%);  
}
```

### Canales Alfa

Es posible que deseemos indicar un color que tenga cierto grado de transparencia, y de esta forma, refleje el contenido, color o imágenes que se encuentren detrás. Hasta ahora solo conocemos la palabra clave **transparent**, que es un color de transparencia total (*totalmente transparente*).

Sin embargo, existe la posibilidad de utilizar los denominados **canales alfa**, que permiten establecer una **transparencia parcial** en determinados colores. Estos se pueden establecer en cualquier formato, salvo en los colores con palabras clave. Vamos a ver como hacerlo en cada caso:

- **Formato RGB:** En lugar de **rgb()** indicamos **rgba()** para establecer que usaremos un canal alfa. Posteriormente, en lugar de establecer 3 parámetros (*rojo, verde, azul*), añadiremos uno más, que será el canal alfa. Dicho canal alfa será un valor (*del 0 al 1 con decimales*) o un porcentaje (*del 0% al 100%*).
- **Formato HSL:** Prácticamente idéntico al anterior. En lugar de **hsl()** indicamos **hsla()**. Añadimos un nuevo valor como canal alfa (*valor o porcentaje*).
- **Formato Hexadecimal:** Es posible indicar (al final) un par adicional que indique el grado de transparencia. Por ejemplo, el color #FF0000 reescrito como #FF000077 se trataría de dicho color, con un grado de transparencia casi del 50% (00 es 0%, 80 es 50%, FF es 100%). (NOTA: Es un borrador y, aunque está soportado actualmente en navegadores como Chrome y Firefox, puede no ser compatible en versiones más antiguas u otros navegadores.)

Veamos algunos ejemplos aplicando grados de transparencia de cada caso:

Ejemplo:

```
div {  
  background-color: rgba(0, 0, 0, 0.5);  
  background-color: rgba(0, 0, 0, 50%);  
  background-color: hsla(180deg, 50%, 25%, 0.75);  
  background-color: hsla(180deg, 50%, 25%, 75%);  
  background-color: #aa44ba80;  
}
```

## 2.2. Unidades

Los tipos de unidades que pueden utilizarse en CSS para indicar un determinado tamaño se pueden clasificar en:

- Unidades absolutas
- Unidades relativas

### Unidades absolutas

Las **unidades absolutas** son un tipo de medida fija que no cambia, que no depende de ningún otro factor. Son ideales para medios no variables como pueden ser los medios impresos, por ejemplo, pero son poco flexibles y adecuados para la web en la actualidad, ya que no tienen la capacidad de adaptarse a diferentes resoluciones o pantallas, que es lo que tendemos a hacer actualmente.

Las diferentes **unidades absolutas** que pueden utilizarse en CSS son las siguientes:

| Unidad | Significado  | Medida aproximada |
|--------|--------------|-------------------|
| in     | Pulgadas     | 1in = 25.4mm      |
| cm     | Centímetros  | 1cm = 10mm        |
| pc     | Picas        | 1pc = 4.23mm      |
| mm     | Milímetros   | 1mm = 1mm         |
| pt     | Puntos       | 1pt = 0.35mm      |
| px     | Píxels       | 1px = 0.26mm      |
| Q      | Cuarto de mm | 1Q = 0.248mm      |

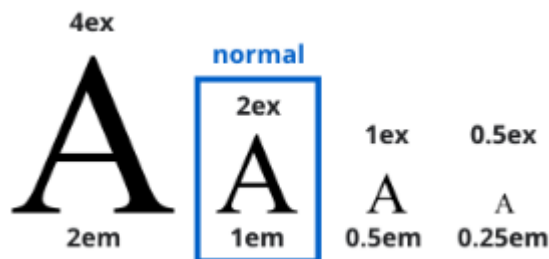
### Unidades relativas

Las **unidades relativas** son un tipo de medida más flexible en CSS. Al contrario que las unidades absolutas, las unidades relativas dependen de algún otro factor (*resolución, densidad de pantalla, etc...*). Tienen una curva de aprendizaje más compleja, pero son las ideales para trabajar en dispositivos con diferentes tamaños, ya que son muy flexibles:

| Unidad | Significado  | Medida aproximada                                   |
|--------|--------------|---|
| em     | «M»          | 1em = tamaño de la fuente del navegador             |
| ex     | «X» (~0.5em) | 1ex = ~ mitad del tamaño de la fuente del navegador |
| ch     | «zero width» | 1ch = ancho del cero                                |
| rem    | «root M»     | 1rem = tamaño fuente raíz                           |
| %      | Porcentaje   | Relativa a herencia                                 |

La unidad **em** se utiliza para hacer referencia al tamaño actual de la fuente del elemento en cuestión. Así, una cantidad de **1em** sería el tamaño actual de la fuente exactamente, y una cantidad de **2em** sería justo el doble. Por otro lado, **1ex** es aproximadamente la mitad del tamaño de la fuente.





Realmente, la medida **ex** está basada en la **altura de la x minúscula**, que es aproximadamente un poco más de la mitad de la fuente actual (depende de la tipografía utilizada), o **ch**, que equivale al tamaño de ancho del 0 de la fuente actual, aunque en la práctica es un tipo de unidad que no suele ser utilizada demasiado.



Una unidad muy interesante y práctica para tipografías es la unidad **rem** (*root em*). Esta unidad es muy cómoda, ya que permite establecer un tamaño para el documento en general (*en el elemento **body***):

Ejemplo:

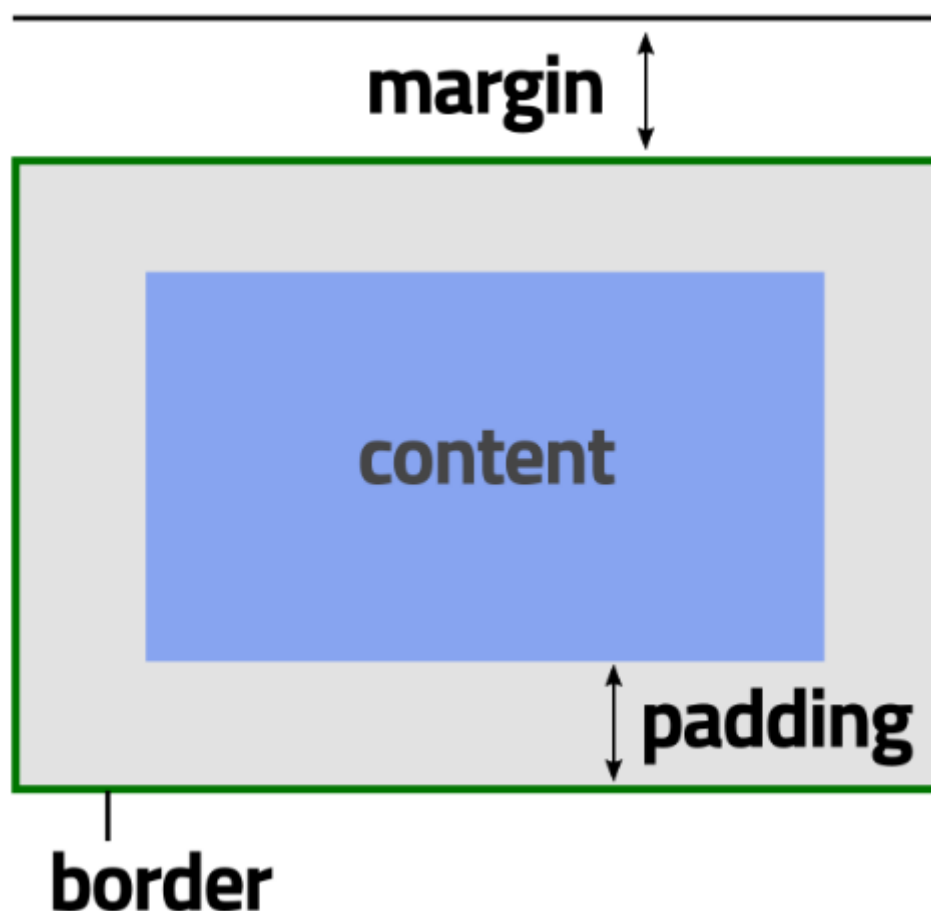
```
body {  
  font-size: 22px;  
}  
  
h1 { font-size: 2rem; }  
h2 { font-size: 1rem; }
```

Posteriormente, podemos ir utilizando la unidad **rem** en ciertas partes del documento. Con esto, estamos indicando el factor de escala (*respecto al tamaño general que indicamos en el **body***). En el ejemplo anterior, los elementos **<h1>** tendrán 44 píxels de tamaño, ya que hemos establecido **2rem**, que significa «el doble que el tamaño general». Por otro lado, los elementos **<h2>** tendrían el mismo tamaño (22 píxels).

Esto nos da una ventaja principal considerable: Si queremos cambiar el tamaño del texto en general, sólo tenemos que cambiar el **font-size** del elemento **body**, puesto que el resto de unidades son factores de escalado y se modificarán todas en consecuencia al cambio del **body**.

### 2.3. Modelo de cajas

La representación básica del **modelo de cajas** es el siguiente:



#### Dimensiones

Para dar tamaños específicos a los diferentes elementos de un documento HTML, necesitaremos asignarles valores a las propiedades **width** (ancho) y **height** (alto). Además, ambas tienen propiedades hermanas para establecer el tamaño máximo y mínimo que pueden alcanzar:

| Propiedad   | Valor                | Significado                                |
|-------------|----------------------|--|
| width:      | <b>auto</b>   TAMAÑO | Tamaño de ancho de un elemento.            |
| max-width:  | <b>none</b>   TAMAÑO | Ancho máximo que puede ocupar un elemento. |
| min-width:  | <b>0</b>   TAMAÑO    | Ancho mínimo que puede ocupar un elemento. |
| height:     | <b>auto</b>   TAMAÑO | Tamaño de alto de un elemento.             |
| max-height: | <b>none</b>   TAMAÑO | Alto máximo que puede ocupar un elemento.  |
| min-height: | <b>0</b>   TAMAÑO    | Alto mínimo que puede ocupar un elemento.  |

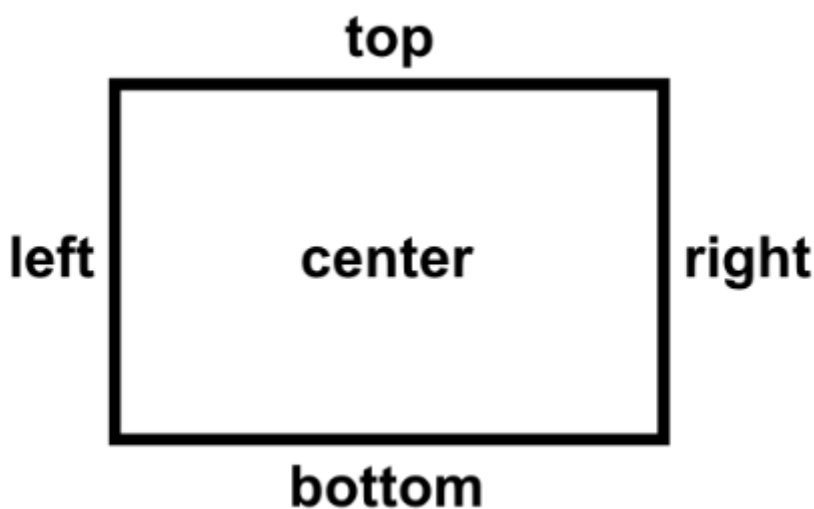
En el caso de utilizar el valor **auto** en las propiedades anteriores, el navegador se encarga de calcular el ancho o alto necesario, dependiendo del contenido del elemento. Las propiedades **min-width** y **min-height**, por defecto tienen valor **0**, mientras que las propiedades **max-width** y **max-height**, por defecto tienen valor **none**.

#### Desbordamiento del contenido

También podemos especificar a través de la propiedad **overflow** el comportamiento que tendrá si el tamaño indicado a un elemento es tan grande que no cabe en él. Los valores que puede tomar son: (LEER tabla)

| Valor          | ¿Qué ocurre si se desborda el contenedor?                                |
|----------------|--|
| <b>visible</b> | Se dibuja el contenido sobrante fuera del contenedor.                    |
| <b>hidden</b>  | Se oculta el contenido sobrante.   |
| <b>scroll</b>  | Se colocan unas barras de desplazamiento (horizontales y verticales).    |
| <b>auto</b>    | El navegador coloca sólo las barras de desplazamiento si son necesarias. |

#### Zonas de un elemento



- **Top:** Se refiere a la parte superior del elemento.
- **Left:** Se refiere a la parte izquierda del elemento.
- **Right:** Se refiere a la parte derecha del elemento.
- **Bottom:** Se refiere a la parte inferior del elemento.
- **Center:** En algunos casos se puede especificar el valor **center** para referirse a la posición central entre dos extremos.



### Márgenes (*margin*) y espaciados (*padding*)

Los márgenes (*margin*) son los espacios que hay entre los bordes del elemento en cuestión y los elementos externos.

| Propiedad      | Valor         | Significado  |
|----------------|---------------|--|
| margin-top:    | auto   TAMAÑO | Separa el elemento por arriba con un margen superior.        |
| margin-left:   | auto   TAMAÑO | Separa el elemento por la izquierda con un margen izquierdo. |
| margin-right:  | auto   TAMAÑO | Separa el elemento por la derecha con un margen derecho.     |
| margin-bottom: | auto   TAMAÑO | Separa el elemento por abajo con un margen inferior.         |

Hay que diferenciar bien los **márgenes** de los **espaciados**, puesto que no son la misma cosa. Los **espaciados** (*padding*) son los espacios que hay entre los bordes del elemento en cuestión y el contenido del elemento. Mientras que los márgenes (*margin*) son los espacios que hay entre los bordes del elemento en cuestión y sus elementos adyacentes.

| Propiedad       | Valor      | Significado  |
|-----------------|------------|--|
| padding-top:    | 0   TAMAÑO | Aplica un relleno interior en el espacio superior de un elemento.  |
| padding-left:   | 0   TAMAÑO | Aplica un relleno interior en el espacio izquierdo de un elemento. |
| padding-right:  | 0   TAMAÑO | Aplica un relleno interior en el espacio derecho de un elemento.   |
| padding-bottom: | 0   TAMAÑO | Aplica un relleno interior en el espacio inferior de un elemento.  |

Por defecto no hay relleno (*el relleno está a cero*), aunque puede modificarse tanto con las propiedades anteriores.

### Atajo: Márgenes y espaciados

| Propiedad | Valores   | Significado  |
|-----------|---|--|
| margin:   | <i>[top, right, bottom, left]</i>                         | 1 parámetro. Aplica el mismo margen a todos los lados.                 |
| margin:   | <i>[top, bottom]</i> <i>[left, right]</i>                 | 2 parámetros. Aplica margen arriba/abajo e izq/dcha.                   |
| margin:   | <i>[top]</i> <i>[right, left]</i> <i>[bottom]</i>         | 3 parámetros. Aplica margen arriba, izq/dcha y abajo.                  |
| margin:   | <i>[top]</i> <i>[right]</i> <i>[bottom]</i> <i>[left]</i> | 4 parámetros. Aplica margen en cada lado, en sentido agujas del reloj. |

Con las propiedades **padding** y **border-width** pasa exactamente lo mismo, actuando en relación a los **espaciados**, en lugar de los márgenes en el primer caso, y en relación al **grosor del borde** de un elemento en el segundo.

## 2.4. Bordes

Las propiedades básicas existentes de los bordes en CSS son las siguientes:

| Propiedad     | Valor                             | Significado   |
|---------------|-----------------------------------|---|
| border-color: | <b>COLOR</b>                      | Especifica el color que se utilizará en el borde.                 |
| border-width: | thin   <b>medium</b>   thick      | Especifica un tamaño predefinido para el grosor del borde.        |
| border-width: | <b>TAMAÑO</b>                     | También se puede indicar el tamaño concreto del grosor del borde. |
| border-style: | none   [ <i>estilo de borde</i> ] | Define el estilo para el borde a utilizar.                        |

En primer lugar, **border-color** establece el color del borde, de la misma forma que lo hicimos en apartados anteriores de colores. En segundo lugar, con **border-width** podemos establecer la anchura o grosor del borde utilizando tanto **palabras clave** predefinidas como un tamaño concreto con cualquier tipo de las **unidades** ya vistas.

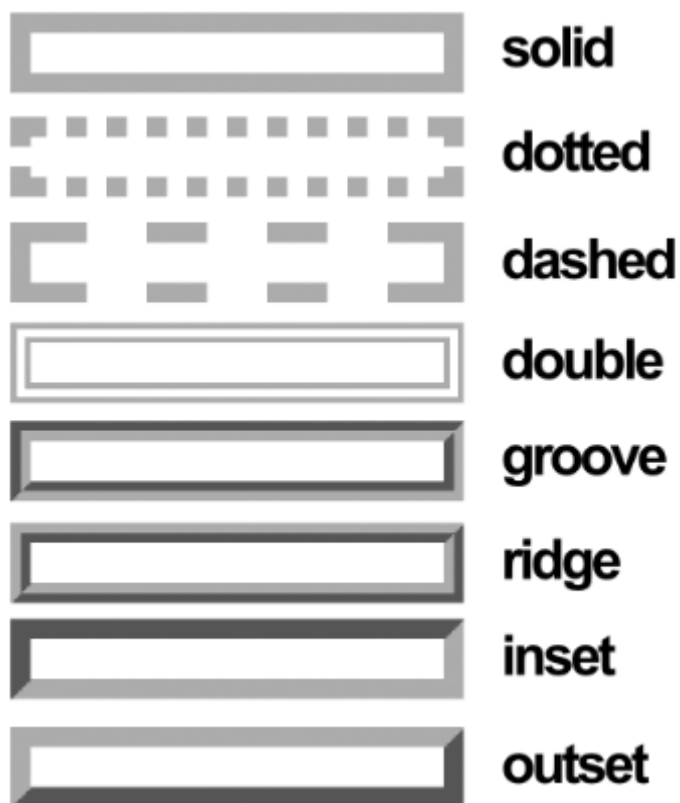
Por último, con **border-style** podemos aplicar un estilo determinado al borde de un elemento. En **estilo de borde** podemos elegir cualquiera de las siguientes opciones:

| Valor  | Descripción  |
|--------|--|
| hidden | Oculto. Idéntico al anterior salvo para conflictos con tablas.             |
| dotted | Establece un borde basado en puntos.                                       |
| dashed | Establece un borde basado en rayas (línea discontinua).                    |
| solid  | Establece un borde sólido (línea continua).                                |
| double | Establece un borde doble (dos líneas continuas).                           |
| groove | Establece un borde biselado con luz desde arriba.                          |
| ridge  | Establece un borde biselado con luz desde abajo. Opuesto a <b>groove</b> . |
| inset  | Establece un borde con profundidad «hacia dentro».                         |
| outset | Establece un borde con profundidad «hacia fuera». Opuesto a <b>inset</b> . |

Ejemplo:

```
div {
  border-color: gray;
  border-width: 1px;
  border-style: dotted;
}
```

Ejemplos de los diferentes estilos de borde utilizando **10 píxels** de grosor y color **gris**:



#### Bordes con varios parámetros

| Propiedad     | Valor                      | Significado  |
|---------------|----------------------------|--|
| border-color: | <i>[c1]</i>                | 1 parámetro. Aplica el <b>COLOR</b> <b>c1</b> a todos los bordes del elemento.               |
|               | <i>[c1] [c2]</i>           | 2 parámetros. <b>c1</b> : borde superior/inferior, <b>c2</b> : laterales.                    |
|               | <i>[c1] [c2] [c3]</i>      | 3 parámetros. <b>c1</b> : borde superior, <b>c2</b> : laterales, <b>c3</b> : borde inferior. |
|               | <i>[c1] [c2] [c3] [c4]</i> | 4 parámetros. Orden de las agujas del reloj, <b>c1</b> : borde superior.                     |

De la misma forma, podemos hacer exactamente lo mismo con las propiedades **border-width** y **border-style**. Teniendo en cuenta esto, tenemos mucha flexibilidad a la hora de especificar esquemas de bordes más complejos:

Ejemplo:

```
div {
  border-color: red blue green;
  border-width: 2px 10px 5px;
  border-style: solid dotted solid;
}
```

En el ejemplo anterior hemos utilizado 3 parámetros, indicando un elemento con borde superior rojo sólido de 2 píxeles de grosor, con borde izquierdo y derecho punteado azul de 10 píxeles de grosor y con un borde inferior verde sólido de 5 píxeles de grosor.

### Atajo: Bordes

Al igual que con otras propiedades CSS, podemos utilizar la propiedad de atajo **border**, con la que podemos hacer un resumen y prescindir de otras propiedades, realizando el proceso de forma más corta:

```
div {  
  border: <width> <style> <color>  
}
```

#### Ejemplo:

```
div {  
  border: 1px solid #000000;  
}
```

Sintaxis equivalente:

```
div {  
  border-width: 1px;  
  border-style: solid;  
  border-color: #000000;  
}
```

### Bordes específicos

Otra forma, quizás más intuitiva, es la de utilizar las propiedades de bordes específicos (*por zonas*) y aplicar estilos combinándolos con la herencia de CSS. Para utilizarlas bastaría con indicarle la zona justo después de **border-**.

#### Ejemplo:

```
div {  
  border-bottom-width: 2px;  
  border-bottom-style: dotted;  
  border-bottom-color: black;  
}
```

Esto dibujaría **sólo un borde inferior** negro de 2 píxeles de grosor y con estilo punteado. Ahora imaginemos que queremos un elemento con todos los bordes en rojo a 5 píxeles de grosor, salvo el borde superior, que lo queremos con un borde de 15 píxeles en color naranja. Podríamos hacer lo siguiente:



```
div {  
  border: 5px solid red;  
  border-top-width: 15px;  
  border-top-color: orange;  
  border-top-style: solid; /* Esta propiedad no es necesaria (se hereda) */  
}
```

El ejemplo anterior conseguiría nuestro objetivo. La primera propiedad establece todos los bordes del elemento, sin embargo, las siguientes propiedades modifican sólo el borde superior, cambiándolo a las características indicadas.

Recuerda que también existen atajos para estas propiedades de bordes en zonas concretas, lo que nos permite simplificar aún más el ejemplo anterior, haciéndolo más fácil de comprender:

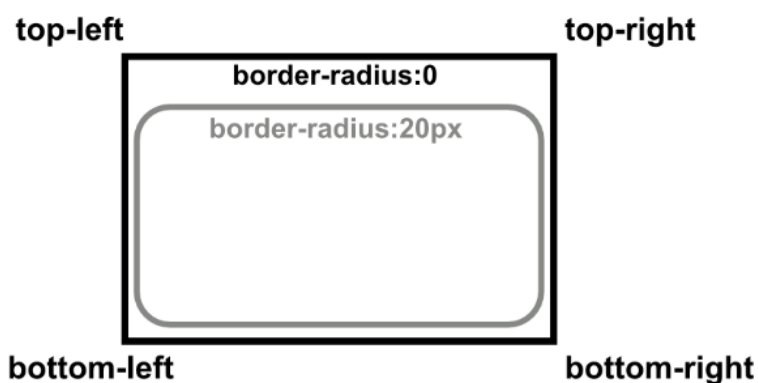
```
div {  
  border: 5px solid red;  
  border-top: 15px solid orange;  
}
```

**Ojo:** Es muy importante entender como se está aplicando la herencia en los ejemplos anteriores, puesto que es una de las características más complejas de dominar de CSS. Por ejemplo, si colocáramos el **border-top** antes del **border**, este último sobrescribiría los valores de **border-top** y no funcionaría de la misma forma.

### Esquinas redondeadas

**CSS3** añade interesantes características en materia de bordes, como la posibilidad de crear **bordes con esquinas redondeadas**, característica que en versiones anteriores de CSS era muy complicado de lograr, necesitando recurrir al apoyo de imágenes gráficas. Por su parte, en **CSS3** es realmente sencillo.

Basta con utilizar la propiedad **border-radius**, con la cual podrás especificar un radio para el borde de las esquinas. Por defecto, este borde es de **tamaño 0**, por lo que no hay borde redondeado. A medida que se aumenta este valor, el borde se redondea más. Una vez llegado a su máximo, no se apreciará ningún cambio.



Hay varias formas de especificar el **radio** de las esquinas:

| Propiedad      | Valor                      | Significado  |
|----------------|----------------------------|--|
| border-radius: | <i>[e1]</i>                | 1 parámetro. Aplica el <b>TAMAÑO</b> de radio <b>e1</b> a todas las esquinas.            |
|                | <i>[e1] [e2]</i>           | 2 parámetros. <b>e1</b> : sup-izq e inf-dcha, <b>e2</b> : sup-dcha e inf-izq.            |
|                | <i>[e1] [e2] [e3]</i>      | 3 parámetros. <b>e1</b> : sup-izq, <b>e2</b> : sup-dcha e inf-izq, <b>e3</b> : inf-dcha. |
|                | <i>[e1] [e2] [e3] [e4]</i> | 4 parámetros. Orden de las agujas del reloj. <b>e1</b> : superior-izquierda.             |

El primer formato, un único parámetro, aplica ese tamaño a todas las esquinas del borde.

El segundo formato, con dos parámetros, aplica el primer valor, **e1**, a las esquinas superior-izquierda e inferior-derecha, y el segundo valor, **e2**, a las esquinas superior-derecha e inferior-izquierda.

En el **tercer formato**, se aplica el parámetro **e1** a la esquina superior-izquierda, el parámetro **e2** a las esquinas superior-derecha e inferior-izquierda y el parámetro **e3** a la esquina inferior-derecha.

Y por último, en el **cuarto formato**, se aplica el tamaño de cada valor a cada esquina por separado, en el sentido de las agujas del reloj. O lo que es lo mismo, **e1** a la esquina superior-izquierda, **e2** a la esquina superior-derecha, **e3** a la esquina inferior-derecha y **e4** a la esquina inferior-izquierda.

A modo de ejemplo teórico, pueden ver un ejemplo de la aplicación de varios formatos:

Ejemplo:

```
div {
  border-radius: 25px;           /* Primer formato */
  border-radius: 25% 50%;       /* Segundo formato */
  border-radius: 50px 25px 10px; /* Tercer formato */
  border-radius: 25px 0 15px 50px; /* Cuarto formato */
}
```

## 2.5. Fondos

### Imágenes de fondo

Para colocar una imagen de fondo se puede utilizar la propiedad **background-image**. En el valor, se introduce el nombre de la imagen o la dirección donde está alojada, siempre rodeado del texto **url()**.

En el caso de que sólo se coloque el nombre de la imagen (*por ejemplo, imagen.jpg*), el navegador buscará la imagen en la misma carpeta donde está el archivo CSS.



| Propiedad         | Valor                    | Significado   |
|-------------------|--------------------------|---|
| background-image: | <b>none</b>              | No utiliza ninguna imagen de fondo.                   |
| background-image: | url( <i>imagen.jpg</i> ) | Usa la imagen de nombre <b>imagen.jpg</b> como fondo. |

**NOTA:** En el caso de que no se encuentre la imagen o el valor de **background-image** se haya establecido a **none**, no se utilizará ninguna imagen de fondo, y en su lugar se mostrará el color establecido con **background-color**.

Por otro lado, una vez establecida una imagen de fondo con **background-image**, se puede personalizar la forma en la que se mostrará dicha imagen mediante propiedades como **background-repeat**, **background-attachment** o **background-position**:

| Propiedad              | Valor            | Significado   |
|------------------------|------------------|---|
| background-repeat:     | <b>repeat</b>    | Repite la imagen de fondo horizontalmente y verticalmente.  |
|                        | repeat-x         | Repite la imagen de fondo sólo horizontalmente (eje x).     |
|                        | repeat-y         | Repite la imagen de fondo sólo verticalmente (eje y).       |
|                        | no-repeat        | La imagen de fondo no se repite.                            |
| background-attachment: | <b>scroll</b>    | Cuando hacemos scroll la imagen de fondo se mueve.          |
|                        | fixed            | Cuando hacemos scroll, la imagen de fondo permanece fija.   |
| background-position:   | <b>POSX</b>      | 1 parámetro. Desplaza la imagen de fondo al punto (x, 50%). |
|                        | <b>POSX POSY</b> | 2 parámetros. Desplaza la imagen de fondo al punto (x,y).   |

### Atajo: Imágenes de fondo

Es posible establecer todas las propiedades anteriores en una sola regla de CSS a modo de atajo, y así ahorrar mucho espacio en escribir las propiedades anteriores por separado. Aunque no es estrictamente obligatorio, se aconseja respetar el siguiente orden (*acostumbrarse a usar el mismo orden es una buena práctica*):

```
div {
  background: <color> <image> <repeat> <attachment> <position>
}
```

### Ejemplo:

```
div {
  background: #FFF url(imagen.jpg) repeat-x scroll top left;
}
```

## 2.6. Tipos de elementos

Cada etiqueta HTML tiene una representación visual en un navegador, lo que habitualmente se suele denominar el **tipo de caja**. Hay dos tipos básicos:

| Valor  | Denominación       | Significado   | Ejemplo                   |
|--------|--------------------|---|---------------------------|
| inline | Elemento en línea  | El elemento se coloca en horizontal (ocupa sólo su contenido).      | <code>&lt;span&gt;</code> |
| block  | Elemento en bloque | El elemento se coloca en vertical (ocupa todo el ancho disponible). | <code>&lt;div&gt;</code>  |

Por defecto, todos los elementos `<div>` son elementos de bloque (*block*) y todos los elementos `<span>` son elementos en línea (*inline*).

Ejemplo:

```
<div>Elemento 1</div>
<div>Elemento 2</div>
<div>Elemento 3</div>
```

A estas etiquetas HTML le vamos a aplicar el siguiente código CSS:

```
div {
  background: blue;
  color: white;
  margin: 1px;
}
```

Se muestra por defecto la etiqueta `<div>` como bloque:

Elemento 1  
Elemento 2  
Elemento 3

Pero si se le aplica a la etiqueta `<div>` la propiedad `display` con el valor `inline` se muestra en línea:

Elemento 1 Elemento 2 Elemento 3

### Ocultar elementos

Mediante la propiedad **display** podemos aplicar un valor **none** y ocultar completamente elementos que no queramos que se muestren, los cuales desaparecen por completo. Es muy útil para hacer desaparecer información cuando el usuario realiza alguna acción, por ejemplo.

| Tipo de caja | Características   |
|--------------|---|
| none         | Hace desaparecer visualmente el elemento, como si no existiera. |





No obstante, también existe una propiedad CSS llamada **visibility** que realiza la misma acción, con la ligera diferencia de que no sólo oculta el elemento, sino que además mantiene un vacío con el mismo tamaño de lo que antes estaba ahí.

Dicha propiedad **visibility** tiene las siguientes opciones:

| Valor           | Significado  |
|-----------------|--|
| <b>visible</b>  | El elemento es visible.  |
| <b>hidden</b>   | El elemento no es visible pero sigue ocupando su espacio.        |
| <b>collapse</b> | Sólo para tablas. El elemento se contrae para no ocupar espacio. |

**Ojo:** Utilizar **visibility: hidden** es muy interesante si queremos que un elemento y su contenido se vuelva invisible, pero siga ocupando su espacio y así evitar que los elementos que le sigan se desplacen hacia arriba, lo que suele ser un comportamiento no deseado en algunos casos.

## 2.7. Fuentes y tipografías

### Propiedades básicas

Las propiedades CSS más básicas para aplicar a cualquier tipo de tipografía son:

| Propiedad    | Valor                            | Significado  |
|--------------|----------------------------------|--|
| font-family: | <u>f</u> uente                   | Indica el nombre de la fuente (tipografía) a utilizar. |
| font-size:   | TAMANO                           | Indica el tamaño de la fuente.                         |
| font-style:  | <b>normal</b>   italic   oblique | Indica el estilo de la fuente.                         |
| font-weight: | <u>p</u> eso                     | Indica el peso (grosor) de la fuente.                  |

### Familia tipográfica

Empezaremos por la más lógica, la propiedad CSS para seleccionar una **familia tipográfica** concreta. Con esta propiedad, denominada **font-family**, podemos seleccionar cualquier tipografía simplemente escribiendo su nombre.

Si dicho nombre está compuesto por varias palabras separadas por un espacio, se aconseja utilizar comillas simples para indicarla (*como se ve en el segundo ejemplo*):

Ejemplo:

```
body {
  font-family: Verdana;
  font-family: 'PT Sans';           /* Otro ejemplo */
}
```



Esta es la forma más básica de indicar una tipografía. Sin embargo, hay que tener en cuenta que estas fuentes **sólo se visualizarán si el usuario las tiene instaladas en su sistema**. En caso contrario, se observarán los textos con otra tipografía que sí esté disponible en el sistema.

Un primer y sencillo paso para paliar (*en parte*) este problema, una buena práctica es añadir varias tipografías alternativas, separadas por comas:

```
div {
  font-family: Vegur, 'PT Sans', Verdana, sans-serif;
}
```

Así pues, el navegador busca la fuente **Vegur** en nuestro sistema, y en el caso de no estar instalada, pasa a buscar la siguiente (*PT Sans*), y así sucesivamente.

Las palabras clave de **fuentes seguras** son las siguientes:

| Fuente            | Significado              | Fuentes de ejemplo          |
|-------------------|--------------------------|-----------------------------|
| <b>serif</b>      | Tipografía con serifa    | Times New Roman, Georgia... |
| <b>sans-serif</b> | Tipografía sin serifa    | Arial, Verdana, Tahoma...   |
| <b>cursive</b>    | Tipografía en cursiva    | Sanvito, Corsiva...         |
| <b>fantasy</b>    | Tipografía decorativa    | Critter, Cottonwood...      |
| <b>monospace</b>  | Tipografía monoespaciada | Courier, Courier New...     |

### Tamaño de la tipografía

Otra de las propiedades más utilizadas con las tipografías es **font-size**, una tipografía que permite especificar el tamaño que tendrá la fuente que vamos a utilizar:

| Propiedad  | Valor   | Tipo de medida                    |
|------------|---|-----------------------------------|
| font-size: | xx-small   x-small   small   <b>medium</b>   large   x-large   xx-large | Absoluta (tamaño predefinido)     |
| font-size: | smaller   larger  | Relativa (más pequeña/más grande) |
| font-size: | <b>TAMAÑO</b>   | Específica (tamaño exacto)        |

Se pueden indicar tres tipos de valores:

- **Medidas absolutas:** Palabras clave como **medium** que representan un tamaño medio (*por defecto*), **small**: tamaño pequeño, **x-small**: tamaño muy pequeño, etc...



- **Medidas relativas:** Palabras clave como **smaller** que representan un tamaño un poco más pequeño que el actual, o **larger** que representa un tamaño un poco más grande que el actual.
- **Medida específica:** Simplemente, indicar píxeles, porcentajes u otra unidad para especificar el tamaño concreto de la tipografía.

### Estilo de la tipografía

A las tipografías elegidas se les puede aplicar ciertos estilos, muy útil para maquetar los textos, como por ejemplo **negrita** o **cursiva** (*italic*). La propiedad que utilizamos es **font-style** y puede tomar los siguientes valores:

| Valor          | Significado  |
|----------------|--|
| <b>normal</b>  | Estilo normal, por defecto. Sin cambios aparentes.                                       |
| <i>italic</i>  | Cursiva. Estilo caracterizado por una ligera inclinación de las letras hacia la derecha. |
| <i>oblique</i> | Oblicua. Idem al anterior, salvo que esta inclinación se realiza de forma artificial.    |

Con la propiedad **font-style** podemos aplicarle estos estilos. En la mayoría de los casos, se aprecia el mismo efecto con los valores **italic** y **oblique**, no obstante, **italic** muestra la versión cursiva de la fuente, específicamente creada por el diseñador de la tipografía, mientras que **oblique** es una representación forzada artificial de una tipografía cursiva.

### Peso de la tipografía

Por otro lado, está el **peso** de la fuente, que no es más que el grosor de la misma. También depende de la fuente elegida, ya que no todas soportan todos los tipos de grosor. De forma similar a como hemos visto hasta ahora, se puede especificar el peso de una fuente mediante tres formas diferentes:

| Propiedad    | Valor                          | Significado                                  |
|--------------|--------------------------------|--|
| font-weight: | <b>normal</b>   <b>bold</b>    | Medidas absolutas (predefinidas)             |
| font-weight: | <b>bolder</b>   <b>lighter</b> | Medidas relativas (dependen de la actual)    |
| font-weight: | <u>peso</u>                    | Medida específica (número del peso concreto) |

- **Valores absolutos:** Palabras claves para indicar el peso de la fuente *normal* y *bold*. **Normal** es el valor por defecto.
- **Valores relativos:** *Bolder* (más gruesa) o *Lighter* (más delgada).
- **Valor numérico:** Un número del **100** (menos gruesa) al **900** (más gruesa). Generalmente se incrementan en valores de 100 en 100.



### Atajo para tipografías

El esquema es el siguiente:

```
div {
  font: <style> <variant> <weight> <size/line-height> <family>;
}
```

Por ejemplo, utilizar la tipografía **Arial**, con la fuente alternativa **Verdana** o una fuente segura sin serifa, a un tamaño de 16 píxeles, con un interlineado de 22 píxeles, un peso de 400, sin utilizar versalitas y con estilo cursiva:

```
div {
  font: italic normal 400 16px/22px Arial, Verdana, Sans-serif;
}
```

## 2.8. Alineaciones y variaciones del texto

Las propiedades CSS que permiten modificar las diferentes alineaciones de los textos en su conjunto son:

| Propiedad     | Valor   | Significado                                    |
|---------------|---|--|
| text-align:   | <b>left</b>   center   right   justify            | Justificación del texto                        |
| text-justify  | <b>auto</b>   inter-word   inter-character   none | Método de justificación de textos              |
| text-overflow | <b>clip</b>   ellipsis   <u>texto</u>             | Comportamiento cuando el texto no cabe «[...]» |

Al igual que existe **text-align** para alinear horizontalmente, también existe la propiedad **vertical-align**, que se encarga de la alineación vertical de un elemento, pudiendo establecer como valor las siguientes opciones:

| Valor           | ¿Cómo hace la alineación?  |
|-----------------|--|
| <b>baseline</b> | La base del elemento con la base del elemento padre.                                     |
| sub             | El elemento como un subíndice.   |
| super           | El elemento como un superíndice.   |
| top             | La parte superior del elemento con la parte superior del elemento más alto de la línea.  |
| middle          | El elemento en la mitad del elemento padre.  |
| bottom          | La parte inferior del elemento con la parte inferior del elemento más bajo de esa línea. |
| text-top        | La parte superior del elemento con la parte superior del texto padre.                    |
| text-bottom     | La parte inferior del elemento con la parte inferior del texto padre.                    |
| <u>tamaño</u>   | Sube o baja un elemento el tamaño o porcentaje especificado.                             |

## Variaciones

Existen propiedades que permiten variar la apariencia de los textos:

| Propiedad        | Valor   | Significado                                   |
|------------------|---|---|
| text-decoration: | <b>none</b>   underline   overline   line-through | Indica el tipo de subrayado (decoración)      |
| text-transform:  | <b>none</b>   capitalize   uppercase   lowercase  | Transforma a mayús/minús o texto capitalizado |

## 2.9. Tablas CSS

Las propiedades específicas para tablas:

| Propiedad        | Valor                      | Significado   |
|------------------|----------------------------|---|
| border-collapse: | <b>separate</b>   collapse | Aplicado sobre una tabla, elimina el espacio de relleno.            |
| border-spacing:  | <b>0</b>   <b>TAMARO</b>   | Amplia el espacio de relleno entre tabla y celdas.                  |
| caption-side:    | <b>top</b>   bottom        | Mueve el elemento <b>&lt;caption&gt;</b> del interior de una tabla. |
| empty-cells:     | <b>show</b>   hide         | Hace desaparecer visualmente una celda vacía (sin texto).           |
| table-layout:    | <b>auto</b>   fixed        | Indica si las celdas deben ajustarse o tener un tamaño fijo.        |

## 2.10. Listas CSS

Las propiedades específicas para las listas:

| Propiedad            | Valor                                 | Significado  |
|----------------------|---------------------------------------|--|
| list-style-image:    | <b>none</b>   url( <i>image.png</i> ) | Especifica una imagen para usar como «punto» o viñeta de ítem. |
| list-style-position: | inside   <b>outside</b>               | Establece o elimina indentación de ítems sobre la lista.       |

| Propiedad        | Valor  | Significado         |
|------------------|--|---------------------|
| list-style-type: | <b>disc</b>   circle   square   none                       | Viñetas sin orden   |
| list-style-type: | decimal   decimal-leading-zero   lower-roman   upper-roman | Viñetas numéricas   |
| list-style-type: | lower-alpha     upper-alpha   lower-greek                  | Viñetas alfabéticas |

Para las listas sin orden **<ul>**:

- **disc**: Un pequeño círculo relleno.
- **circle**: Un círculo vacío.
- **square**: Un cuadrado relleno.
- **none**: No muestra ninguna marca a la izquierda de los ítems.



Para las listas numeradas **<ol>**, se pueden utilizar valores como:

- **decimal**: Numeración decimal: 1, 2, 3, 4, 5...
- **decimal-leading-zero**: Numeración decimal con ceros: 01, 02, 03, 04, 05...
- **lower-roman**: Números romanos en minúsculas: i, ii, iii, iv, v...
- **upper-roman**: Números romanos en mayúsculas: I, II, III, IV, V...
- **lower-alpha / lower-latin**: Minúsculas: a, b, c, d, e...
- **upper-alpha / upper-latin**: Mayúsculas: A, B, C, D, E...
- Otras menos utilizadas generalmente, como **lower-greek** (letras griegas minúsculas), **arabic-indic** (números árabes), **upper-armenian / armenian** (letras armenias en mayúsculas), **lower-armenian** (letras armenias en minúsculas) o **georgian** (letras georgianas).

#### Atajo: Listas

Es posible utilizar la propiedad de atajo **list-style** para especificar varias propiedades en una sola. El orden aconsejado es el siguiente:

```
div {  
  list-style: <type> <position> <image>  
}
```

## 3. Selectores CSS

### 3.1. Selectores CSS básicos

Ejemplo: Sintaxis más básica para especificar un selector (en este caso div)

```
div {  
    background-color: red;  
}
```

Esquema general (simplificado) selectores:

```
selector #id .clase :p  
  
propiedad : valor ;  
propiedad : valor  
  
}
```

#### *Selectores de elementos*

La forma más básica de seleccionar elementos en CSS es indicar el elemento al cuál queremos aplicarle los estilos.

Ejemplo:

```
strong {  
    color: red;  
}
```

#### *Selección de elementos por ID (únicos)*

Todos los elementos HTML pueden tener un atributo **id** con un valor concreto. Este valor será el nombre que le daremos a la etiqueta. **Sólo debe haber una etiqueta con el mismo ID** por documento.

Ejemplo: Uso id

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Documento de ejemplo</title>  
  </head>  
  <body>  
    <div id="saludo">  
      <p>¡Hola, visitante! ¡Bienvenido a esta página! </p>  
    </div>  
  
    <div id="main">  
      <p>En esta página encontrarás los siguientes temas:</p>  
    </div>  
  </body>  
</html>
```



En CSS se hace referencia a los IDs con el símbolo #. Si en el documento HTML anterior nos interesara destacar con colores sólo la primera capa, lo haríamos así:

```
#saludo {  
  background-color: blue;  
  color: white;  
}
```

### *Selección de elementos por clases*

El valor del atributo **class**, a diferencia del valor del atributo **id**, puede ser utilizado en más de un elemento de tu documento HTML. Cuando se tienen que aplicar los mismos estilos a diferentes elementos resulta muy útil, ya que nos permite reducir las líneas de código en nuestro archivo CSS.

Ejemplo: Uso class

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Documento de ejemplo</title>  
  </head>  
  <body>  
    <div id="main">  
      <p>Escoge uno de los siguientes botones:</p>  
  
      <button class="estandar">Opción 1</button>  
      <button class="estandar">Opción 2</button>  
      <button class="estandar">Opción 3</button>  
      <button class="back">Volver</button>  
    </div>  
  </body>  
</html>
```

En CSS se hace referencia a las clases con un punto. Veamos cómo les daríamos estilo CSS a los botones del documento HTML del ejemplo anterior:

```
.estandar {  
  background-color: green;  
  color: white;  
}  
.back {  
  background-color: orange;  
  color: white;  
}
```



Además, en el caso de las clases, podemos incluso diferenciar el tipo de elemento del que se trata. Gracias a eso podríamos, por ejemplo, utilizar una clase **estandar** para dar estilo a botones, y una clase **estandar** para dar estilo a párrafos:

```
button.estandar {  
    background-color: green;  
    color: white;  
}  
p.estandar {  
    color: red;  
}
```

**NOTA:** Se pueden utilizar varias clases en un mismo elemento HTML, simplemente separando por espacios dentro del atributo **class**. De esta forma, dicho elemento tendrá los estilos de cada una de las clases indicadas.

Ejemplo:

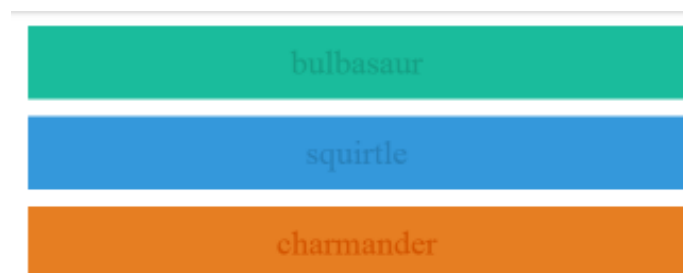
Código HTML

```
<div class="pkmn planta">bulbasaur</div>  
<div class="pkmn agua">squirtle</div>  
<div class="pkmn fuego">charmander</div>
```

Código CSS

```
.pkmn {  
    margin: 1rem;  
    padding: 1rem;  
    text-align: center;  
    font-size: 2rem;  
}  
.planta {  
    background: #1abc9c;  
    color: #16a085;  
}  
.agua {  
    background: #3498db;  
    color: #2980b9;  
}  
.fuego {  
    background: #e67e22;  
    color: #d35400;  
}
```

Salida



### 3.2. Selectores CSS avanzados

Existe un amplio abanico de métodos para seleccionar elementos dependiendo de la estructura del documento HTML denominados **combinadores CSS**:

| Nombre                     | Ejemplo                         | Significado  |
|----------------------------|---------------------------------|--|
| Agrupación de selectores   | <code>p, div { }</code>         | Se aplican estilos a varios elementos.             |
| Selector descendiente      | <code>#page div { }</code>      | Se aplican estilos a elementos dentro de otros.    |
| Selector hijo              | <code>#page &gt; div { }</code> | Se aplican estilos a elementos hijos directos.     |
| Selector hermano adyacente | <code>div + div { }</code>      | Se aplican estilos a elementos que siguen a otros. |
| Selector hermano general   | <code>div ~ div { }</code>      | Se aplican estilos a elementos al mismo nivel.     |
| Selector universal         | <code>#page * { }</code>        | Se aplican estilos a todos los elementos.          |

#### Agrupación de selectores

En muchas ocasiones nos ocurrirá que tenemos varios bloques CSS diferentes con los mismos estilos exactos:

```
strong {
  border-color: red;
  background: white;
}

span {
  border-color: red;
  background: white;
}

img {
  border-color: red;
  background: white;
}
```

En estos casos, se puede hacer uso de la agrupación de selectores para simplificar el documento CSS:

```
strong, span, img {
  border-color: red;
  background: white;
}
```

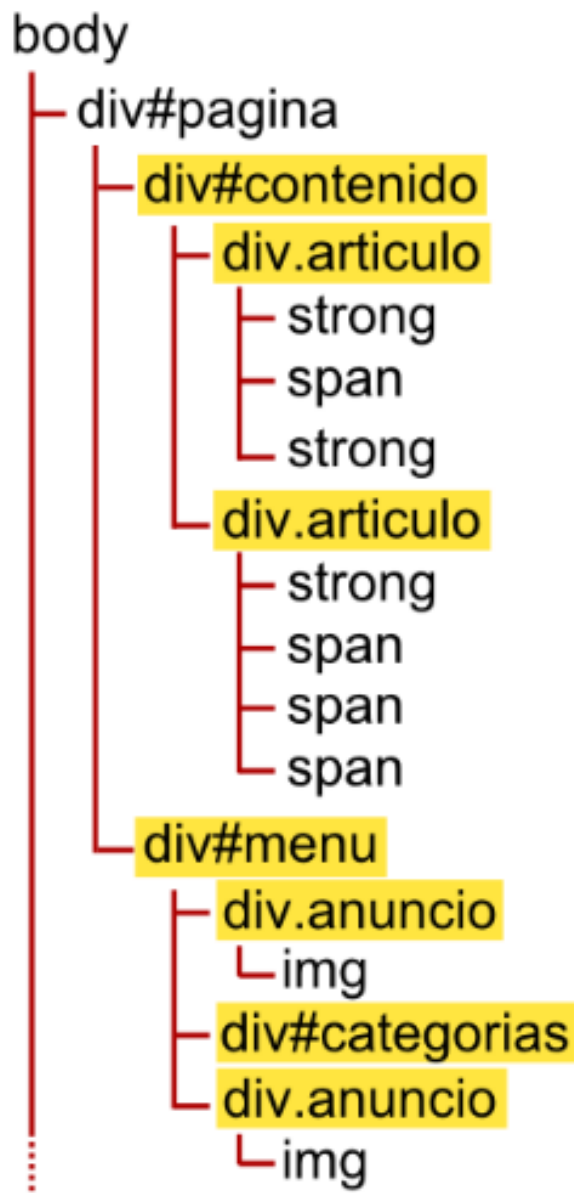
### Selector descendiente

Para seleccionar ciertos elementos que están dentro de otros elementos se utiliza el selector descendiente. Su sintaxis se basa en colocar los elementos uno a continuación de otro, separado por un espacio:

```
div#pagina div {  
  background-color: blue;  
}
```

En el ejemplo anterior, aplicamos los estilos CSS (*color azul de fondo*) a todos los elementos **div** que estén dentro de un **div** con ID **pagina**..

## CSS: **div#pagina div**

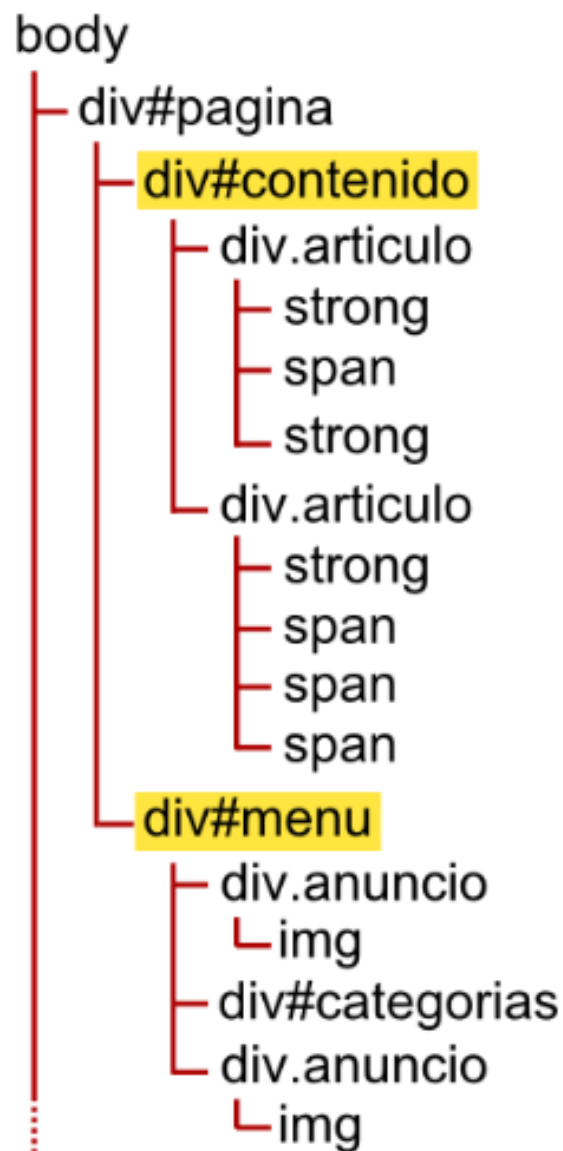


### Selector hijo

Aunque el **selector descendiente** es bastante interesante, nos puede interesar hacer la misma operación, pero en lugar de seleccionar todos los elementos descendientes, seleccionar sólo los descendientes directos del elemento, descartando así nietos y sucesivos.

```
div#pagina > div {  
  background-color: blue;  
}
```

## CSS: *div#pagina > div*

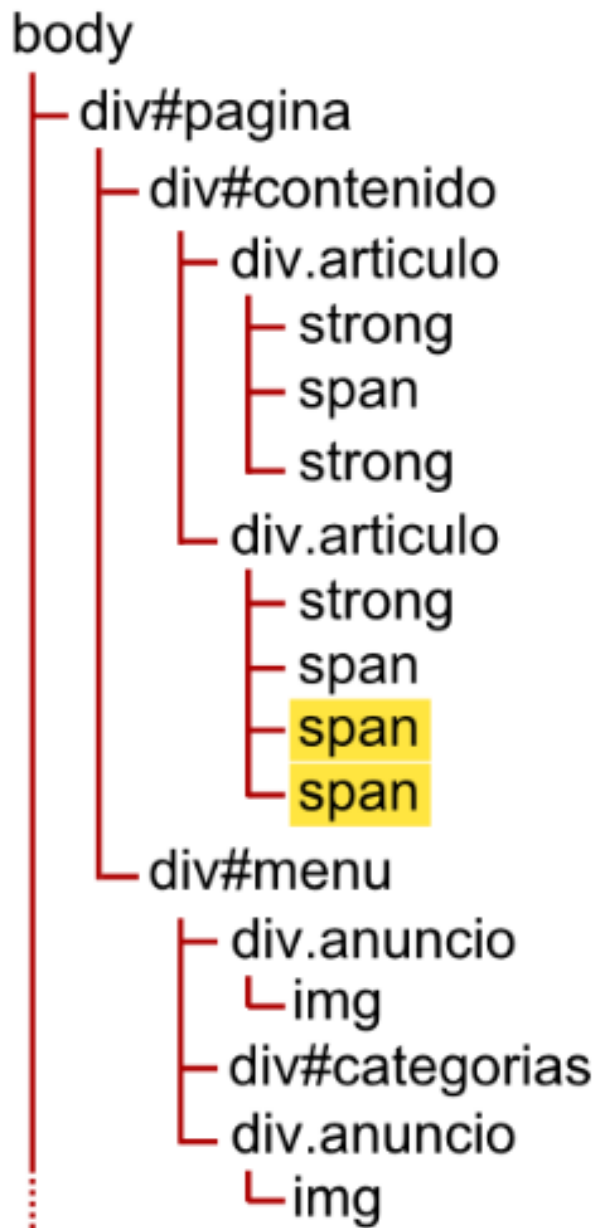


### Selector hermano adyacente

Mediante el **selector hermano adyacente** se pueden seleccionar aquellos elementos hermanos que están seguidos el uno de otro (*en el mismo nivel*).

```
div.articulo span + span {  
  color: blue;  
}
```

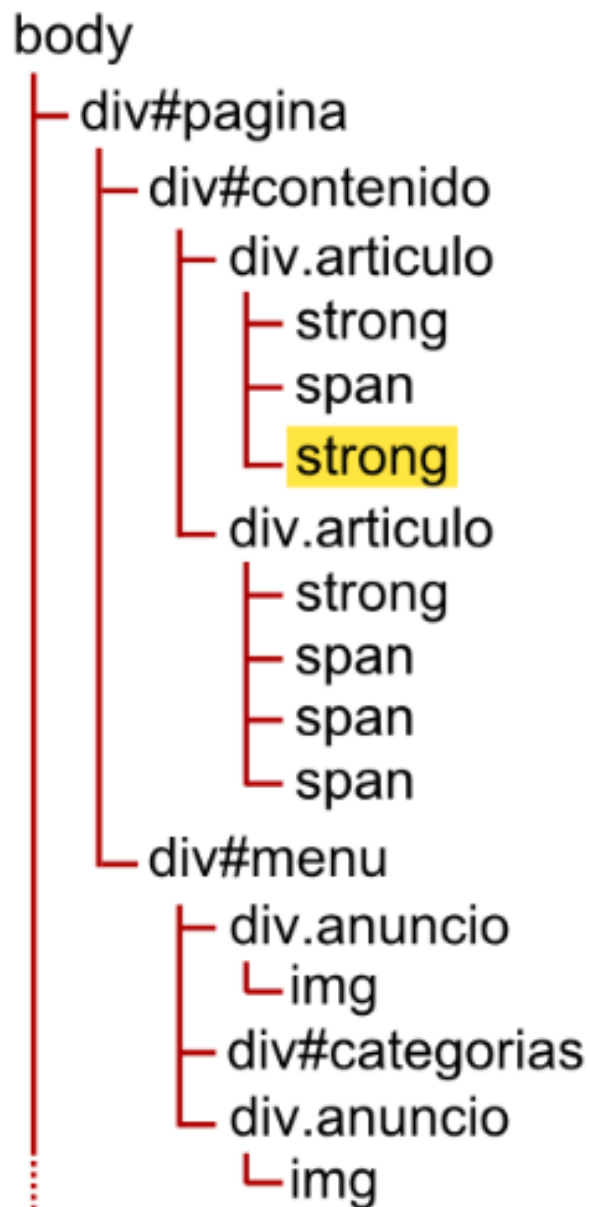
## CSS: *div.articulo span + span*



### Selector hermano general

Para seleccionar todos los hermanos en general, sin necesidad de que sean adyacentes se puede conseguir con el **selector hermano general**, que se simboliza con el símbolo ~.

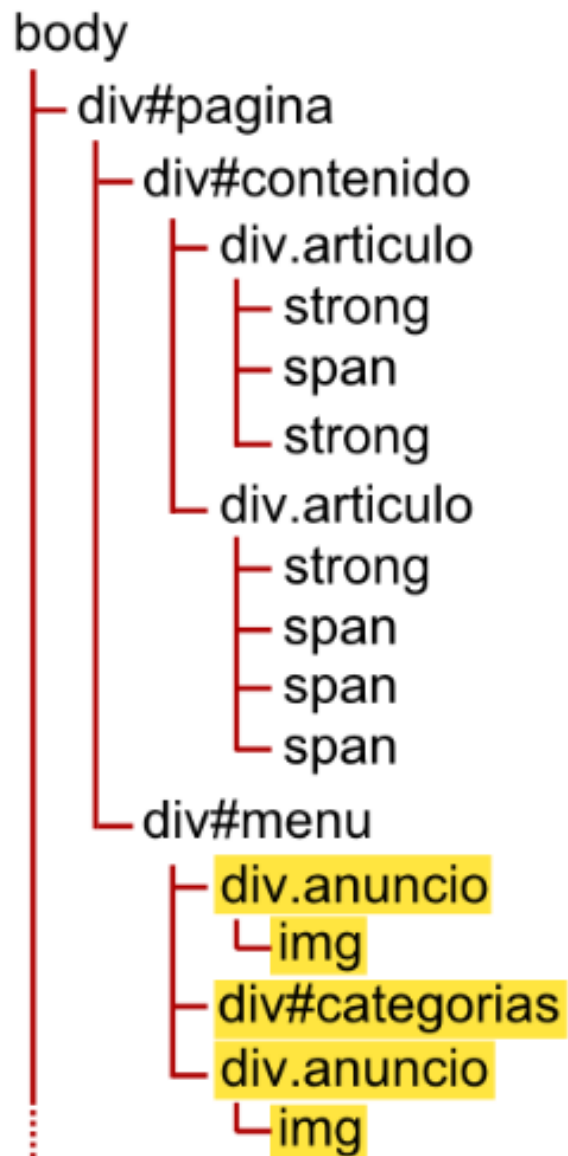
## CSS: *div.articulo strong ~ strong*



### Selector universal

El **selector universal** se simboliza con un asterisco (\*) y es la forma de aplicar ciertos estilos en **todos** los elementos HTML de un documento.

## CSS: `div#menu *`



### 3.3. Pseudoclases CSS

Las **pseudoclases** se utilizan para hacer referencia a ciertos comportamientos de los diferentes elementos del documento HTML.

Esquema general de sintaxis de CSS:

```
selector #id .clase :pseudoclase ::pseudoelemento [atributo] {  
    propiedad : valor ;  
    propiedad : valor  
}
```

Las pseudoclases se definen añadiendo **dos puntos** antes de la pseudoclase concreta. En el caso de existir selectores de etiqueta, id o clases, estas se escribirían a su izquierda.

#### Enlaces

Existen algunas pseudoclases orientadas a los enlaces o hipervínculos. En este caso, permiten cambiar los estilos dependiendo del comportamiento del enlace:

| Pseudoclase           | Descripción   |
|-----------------------|---|
| <code>:link</code>    | Aplica estilos cuando el enlace no ha sido visitado todavía.    |
| <code>:visited</code> | Aplica estilos cuando el enlace ha sido visitado anteriormente. |

Ejemplo: Selector para los enlaces que aún no han sido visitados

```
a:link {  
    color: green;  
    font-weight: bold  
}
```

Ejemplo: Selector para los enlaces que ya han sido visitados.

```
a:visited {  
    color: purple;  
    font-weight: bold  
}
```



### Ratón

| Pseudoclase          | Descripción   |
|----------------------|---|
| <code>:hover</code>  | Aplica estilos cuando pasamos el ratón sobre un elemento. |
| <code>:active</code> | Aplica estilos cuando estamos pulsando sobre el elemento. |

Ejemplo: Aplica estilos a un elemento cuando el usuario está pasando el ratón sobre él

```
a:hover {  
  background-color: cyan;  
  padding: 2px  
}
```

Ejemplo: Aplica estilos a los elementos que se encuentran activos (usuario está pulsando de forma activa con el ratón)

```
a:active {  
  border: 2px solid #FF0000;  
  padding: 2px  
}
```

### Interacción del usuario (orientadas a los campos de formularios)

| Pseudoclase           | Descripción   |
|-----------------------|---|
| <code>:focus</code>   | Aplica estilos cuando el elemento tiene el foco.    |
| <code>:checked</code> | Aplica estilos cuando la casilla está seleccionada. |

Ejemplo:

```
input:focus {  
  border: 2px dotted #444  
}
```

**Nota:** Aunque estas pseudoclases suelen utilizarse con elementos de formularios como `<input>`, pueden utilizarse con otros elementos, como por ejemplo los enlaces `<a>`.

### 3.4. Pseudoclases CSS avanzadas

Existen varias pseudoclases que permiten hacer referencias a los elementos del documento HTML según su **posición y estructura de los elementos hijos**. A continuación muestro un pequeño resumen de estas pseudoclases:

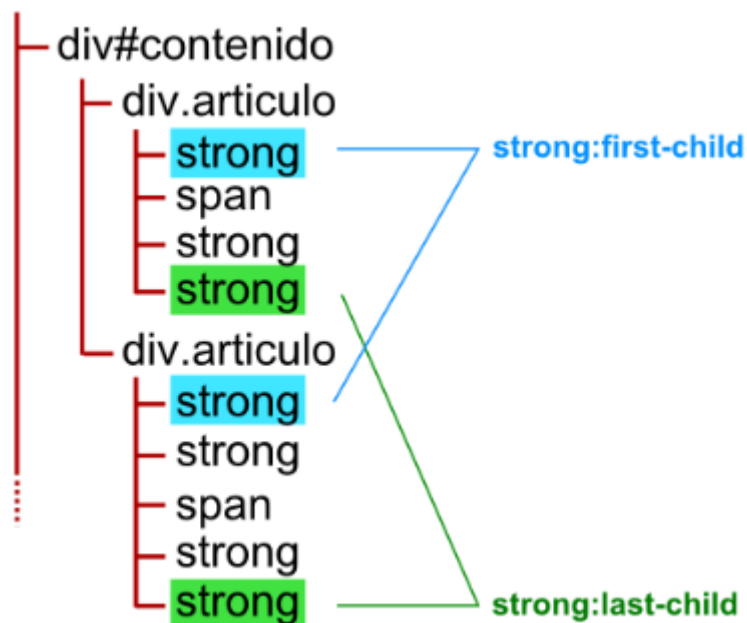
| Pseudoclase                     | Descripción   |
|---------------------------------|---|
| <code>:first-child</code>       | Primer elemento hijo (de cualquier tipo).                     |
| <code>:last-child</code>        | Último elemento hijo (de cualquier tipo).                     |
| <code>:nth-child(n)</code>      | N-elemento hijo (de cualquier tipo).                          |
| <code>:nth-last-child(n)</code> | N-elemento hijo (de cualquier tipo) partiendo desde el final. |

Para ello, volvamos a utilizar una estructura en forma de árbol para ver cómodamente la ubicación de cada uno de los elementos.

Las dos primeras pseudoclases, **`:first-child`** y **`:last-child`** hacen referencia a los primeros y últimos elementos (*al mismo nivel*) respectivamente.

```
strong:first-child {
  background-color: cyan;
}

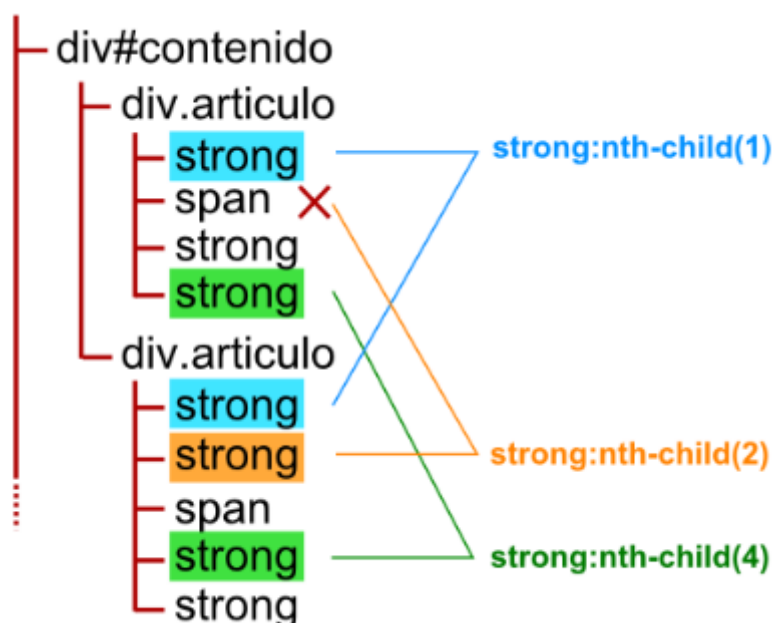
strong:last-child {
  background-color: green;
}
```



Sin embargo, si no queremos quedarnos en los primeros o últimos elementos y necesitamos más potencia para elegir, podemos hacer uso de la pseudoclase **`:nth-child(A)`**, que permite especificar el elemento deseado, simplemente estableciendo su número en el parámetro **A**:

| Número                              | Equivalente a la pseudoclase       | Significado   |
|-------------------------------------|------------------------------------|---|
| <code>strong:nth-child(1)</code>    | <code>strong:first-child {}</code> | Primer elemento hijo, que además es un <code>&lt;strong&gt;</code>  |
| <code>strong:nth-child(2)</code>    |                                    | Segundo elemento hijo, que además es un <code>&lt;strong&gt;</code> |
| <code>strong:nth-child(3)</code>    |                                    | Tercer elemento hijo, que además es un <code>&lt;strong&gt;</code>  |
| <code>strong:nth-child(n)</code>    |                                    | Todos los elementos hijos que son <code>&lt;strong&gt;</code>       |
| <code>strong:nth-child(2n)</code>   |                                    | Todos los elementos hijos pares <code>&lt;strong&gt;</code>         |
| <code>strong:nth-child(2n-1)</code> |                                    | Todos los elementos hijos impares <code>&lt;strong&gt;</code>       |

Veamos además un ejemplo gráfico:



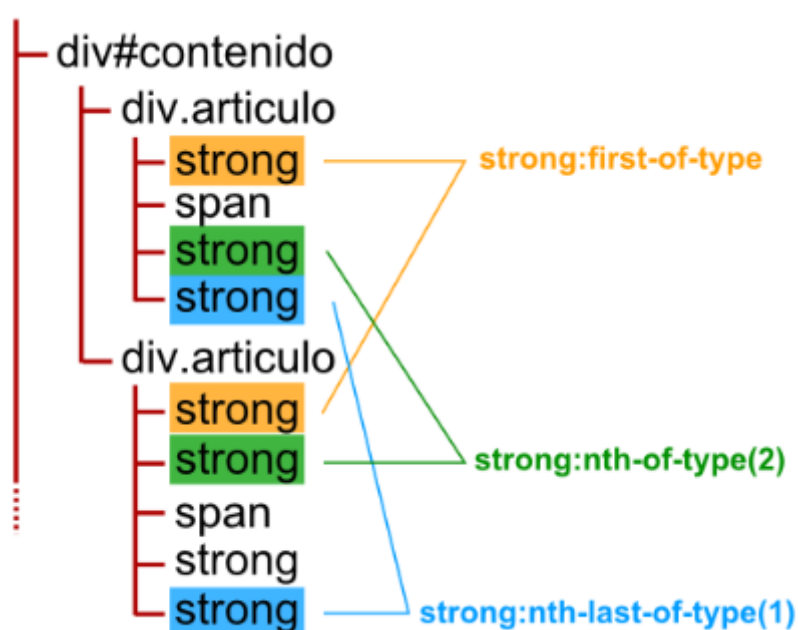
Como se aprecia en el ejemplo, en el caso `:nth-child(2)` se puede ver como el segundo elemento lo ocupa un elemento `span`, por lo que sólo se selecciona el elemento `strong` del segundo caso, donde si existe.

### Elementos del mismo tipo

En los casos anteriores, seleccionamos elementos independientemente de que elemento sea. Simplemente, hacemos caso a la posición donde está ubicado. Si queremos hacer referencia sólo a elementos del mismo tipo, utilizaremos los selectores siguientes, análogos a los anteriores, pero haciendo referencia sólo a elementos del mismo tipo:

| Pseudoclase                 | Descripción  |
|-----------------------------|--|
| <b>:first-of-type</b>       | Primer elemento hijo (de su mismo tipo).                     |
| <b>:last-of-type</b>        | Último elemento hijo (de su mismo tipo).                     |
| <b>:nth-of-type(n)</b>      | N-elemento hijo (de su mismo tipo).                          |
| <b>:nth-last-of-type(n)</b> | N-elemento hijo (de su mismo tipo) partiendo desde el final. |

Las pseudoclases **:first-of-type** y **:last-of-type** son las equivalentes a **:first-child** y **:last-child** pero sólo teniendo en cuenta elementos del mismo tipo. Por otro lado, la pseudoclase **:nth-of-type(A)** es la equivalente a **:nth-child(A)** y **:nth-last-of-type(A)** es la equivalente a **:nth-last-child(A)**. Veamos un ejemplo sobre el ejercicio anterior:



En este ejemplo, se puede ver como **:nth-of-type(2)** selecciona el segundo elemento **strong** en ambos casos, a pesar de que en el primero ocupa la tercera posición. En este caso se selecciona porque es el segundo elemento de su mismo tipo (**<strong>**).

Por otro lado, **:nth-last-of-type(A)** hace una selección de forma inversa, empezando por el último elemento.

### Hijos únicos

Existen también varias pseudoclases para la gestión de hijos únicos. Son las siguientes:

| Pseudoclase          | Descripción                                     |
|----------------------|---|
| <b>:only-child</b>   | Elemento que es hijo único (de cualquier tipo). |
| <b>:only-of-type</b> | Elemento que es hijo único (de su mismo tipo).  |
| <b>:empty</b>        | Elemento vacío (sin hijos, ni texto).           |

La propiedad **:only-child** nos proporciona un método para aplicar estilo a aquellos elementos que sean el único hijo de su elemento padre. Además, como ha ocurrido anteriormente, también existe la pseudoclase **:only-of-type** que es equivalente al anterior pero sólo para elementos del mismo tipo, es decir, que puede ser que no sea el único elemento hijo, pero sí el único de su tipo.

Muy relacionada está también la pseudoclase **:empty**, que permite seleccionar los elementos vacíos que no contengan absolutamente nada (*o espacios o etiquetas de comentarios HTML*).

### 3.5. Pseudoelementos CSS

Al igual que las pseudoclases, los pseudoelementos son otra de las características de CSS que permiten hacer referencias a comportamientos virtuales no tangibles que no tienen relación directa con el contenido del documento en sí.

Recordemos la sintaxis de los pseudoelementos, que está precedida de **dos puntos dobles (::)** para diferenciarlos de las pseudoclases, las cuales sólo tienen **dos puntos (:)**. No obstante, este cambio surgió a partir de CSS3, por lo que aún es frecuente ver pseudoelementos con la sintaxis de pseudoclase con un solo **:**.

```
selector #id .clase :pseudoclase ::pseudoelemento [atributo] {
  propiedad : valor ;
  propiedad : valor
}
```

#### Generación de contenido

Antes de continuar, hay que saber que en este apartado de pseudoelementos CSS es muy común utilizar algunas propiedades interesantes para generar contenido de forma automática, como es el caso de la propiedad **content**. Esta útil propiedad se combina con los pseudoelementos **::after** o **::before**, para añadir información antes o después de un elemento:

| Pseudoelemento  | Descripción  |
|-----------------|--|
| <b>::before</b> | Aplica los estilos <b>antes</b> del elemento indicado.   |
| <b>::after</b>  | Aplica los estilos <b>después</b> del elemento indicado. |

La propiedad **content** admite parámetros de diverso tipo, incluso concatenando información mediante espacios. Podemos utilizar tres tipos de contenido:

| Valor                   | Descripción                                 | Ejemplo  |
|-------------------------|---|--|
| " <i>string</i> "       | Añade el contenido de texto indicado.       | <b>content: "Contenido: ";</b>                 |
| attr( <i>atributo</i> ) | Añade el valor del atributo indicado.       | <b>content: attr(href);</b>                    |
| url( <i>URL</i> )       | Añade la imagen indicada en la <b>URL</b> . | <b>content: url(http://page.com/icon.png);</b> |

Por otro lado, los pseudoelementos **::before** y **::after** permiten hacer referencia a la ubicación **justo antes** y **justo después** (*respectivamente*) del elemento en cuestión.

```
q::before {  
  content: "«";  
  color: #888;  
}  
  
q::after {  
  content: "»";  
  color: #888;  
}
```

### Primera letra y primera línea

También existen pseudoelementos con los que podemos hacer referencia a la **primera letra** de un texto. Para ello utilizamos el pseudoelemento **::first-letter**, así como el pseudoelemento **::first-line** si queremos hacer referencia a la **primera línea** de un texto. De esta forma, podemos dar estilo a esas secciones concretas del texto:

| Pseudoelemento        | Descripción                                       |
|-----------------------|---|
| <b>::first-letter</b> | Aplica los estilos en la primera letra del texto. |
| <b>::first-line</b>   | Aplica los estilos en la primera línea del texto. |

Veamos un ejemplo en acción sobre un párrafo de texto:

```
p {  
  color: #333;  
  font-family: Verdana, sans-serif;  
  font-size: 16px;  
}  
  
p::first-letter {  
  color: black;  
  font-family: 'Times New Roman', serif;  
  font-size: 42px;  
}  
  
p::first-line {  
  color: #999;  
}
```

El diagrama muestra un párrafo de texto: "Esto es un minúsculo texto de ejemplo, para mostrar las ventajas del CSS en la maquetación de texto." Las etiquetas **::first-letter** y **::first-line** están conectadas por líneas a la primera letra 'E' y a la primera línea del texto, respectivamente. La primera letra 'E' está en una fuente serif y de mayor tamaño, mientras que el resto del texto es en una fuente sans-serif.

### 3.6 Atributos CSS

También se pueden aplicar estilos dependiendo del contenido de ciertos atributos de los elementos HTML. Existen varios formatos de atributos, derivados de las expresiones regulares

| Atributo           | Descripción  |
|--------------------|--|
| [href]             | El atributo <b>href</b> existe en la etiqueta.   |
| [href="#"]         | El atributo <b>href</b> existe y su valor es igual al texto <b>#</b> .                       |
| [href\*="emezeta"] | El atributo <b>href</b> existe y su valor contiene el texto <b>emezeta</b> .                 |
| [href^="https://"] | El atributo <b>href</b> existe y su valor comienza por <b>https://</b> .                     |
| [href\$=".pdf"]    | El atributo <b>href</b> existe y su valor termina por <b>.pdf</b> (es un enlace a un PDF).   |
| [class~="emezeta"] | El atributo <b>class</b> contiene una lista de valores, que contiene <b>emezeta</b> .        |
| [lang ="es"]       | El atributo <b>lang</b> contiene una lista de valores, donde alguno empieza por <b>es-</b> . |

Ejemplo: Aplica estilos a los elementos con el atributo style

```
[style] {  
  background: green;  
}
```

Ejemplo: Aplica estilos a los enlaces cuyo atributo rel tenga el valor "nofollow"

```
a[rel="nofollow"] {  
  background: red;  
}
```

## 4. Cascada en CSS

Además de la herencia, otro de los conceptos más importantes de CSS es el concepto de **cascada**. Este concepto es un poco más avanzado, por lo que se debe conocer bien el tema de **selectores CSS** y dominar algo de CSS para comprenderlo en su totalidad.

Supongamos que nos encontramos ante el siguiente caso, ¿cuál de las dos reglas prevalece?:

Ejemplo:

```
div {  
    color: red;  
    padding: 8px  
}  
  
div {  
    color: blue;  
    background-color: grey  
}
```

Prevalece siempre la última regla definida, la cual **mezcla y sobrescribe** las propiedades anteriores.

En el caso anterior, el resultado final sería el siguiente:

```
div {  
    color: blue;           /* Se sobrescribe la última */  
    padding: 8px;  
    background-color: grey  
}
```

Supongamos el siguiente caso:

Ejemplo:

```
div {  
    color: red;  
}  
  
#nombre {  
    color: blue;  
}  
  
.clase {  
    color: green;  
}
```

Tenemos un elemento HTML `<div id="nombre" class="clase">` ¿Cómo sabe CSS que estilo aplicar? ¿Cuál tiene prioridad sobre los demás?



Para saber que bloque de estilos tiene prioridad, CSS analiza *por orden* los siguientes conceptos clave del código CSS: su **especificidad** y su **orden**. Veamos en que se basa cada uno de ellos.

### Especificidad

Para determinar la especificidad de los selectores CSS se sigue un cálculo matemático basado en 4 componentes: **A,B,C,D**.

| Componente   | Descripción   |
|--------------|---|
| Componente A | Número de estilos aplicados mediante un atributo <b>style</b> .                                     |
| Componente B | Número de veces que aparece un <b>id</b> en el selector.  |
| Componente C | Número de veces que aparece una <b>clase</b> , <b>pseudoclase</b> o <b>atributo</b> en el selector. |
| Componente D | Número de veces que aparece un <b>elemento</b> o un <b>pseudoelementos</b> en el selector.          |

Para saber si un bloque de CSS es más específico que otro sólo hay que calcular sus componentes.

#### Ejemplos:

```
div { ... } /* Especificidad: 0,0,0,1 */
div div { ... } /* Especificidad: 0,0,0,2 */
#pagina div { ... } /* Especificidad: 0,1,0,1 */
#pagina div:hover { ... } /* Especificidad: 0,1,1,1 */
#pagina div:hover a { ... } /* Especificidad: 0,1,1,2 */
#pagina .sel:hover > a { ... } /* Especificidad: 0,1,2,1 */
```

### Orden

En CSS, es posible crear múltiples reglas CSS para definir un mismo concepto. En este caso, la que prevalece ante todas las demás depende de ciertos factores, como es la «*altura*» a la que está colocada la regla:

- El **CSS embebido** en un elemento HTML es el que tiene mayor precedencia, por lo que siempre será el que tenga prioridad sobre otras reglas CSS.
- En segundo lugar, el **CSS interno** definido a través de bloques **style** en el propio documento HTML será el siguiente a tener en cuenta en orden de prioridad.
- Por último, los documentos **CSS externos** son la tercera opción de prioridad a la hora de tomar en cuenta las reglas CSS.

Teniendo esto en cuenta, hay que recordar que las propiedades que prevalecerán serán las que estén en último lugar, siempre respetando la prioridad de la lista anterior.

## 5. “Chuleta” de CSS

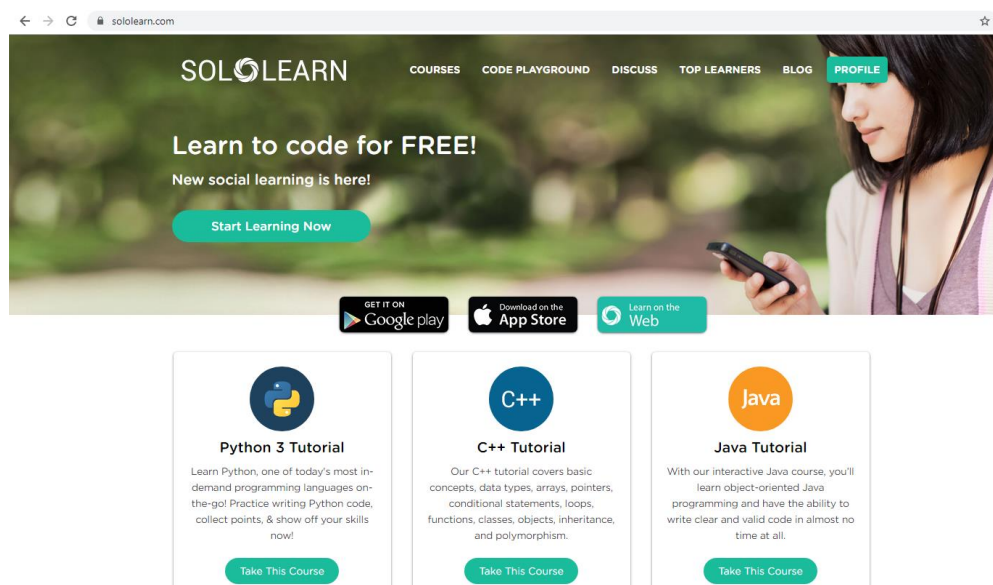
Aquí puedes encontrar (<https://lenguajecss.com/>) una hoja de referencia en formato PDF..



## 6. Sololearn

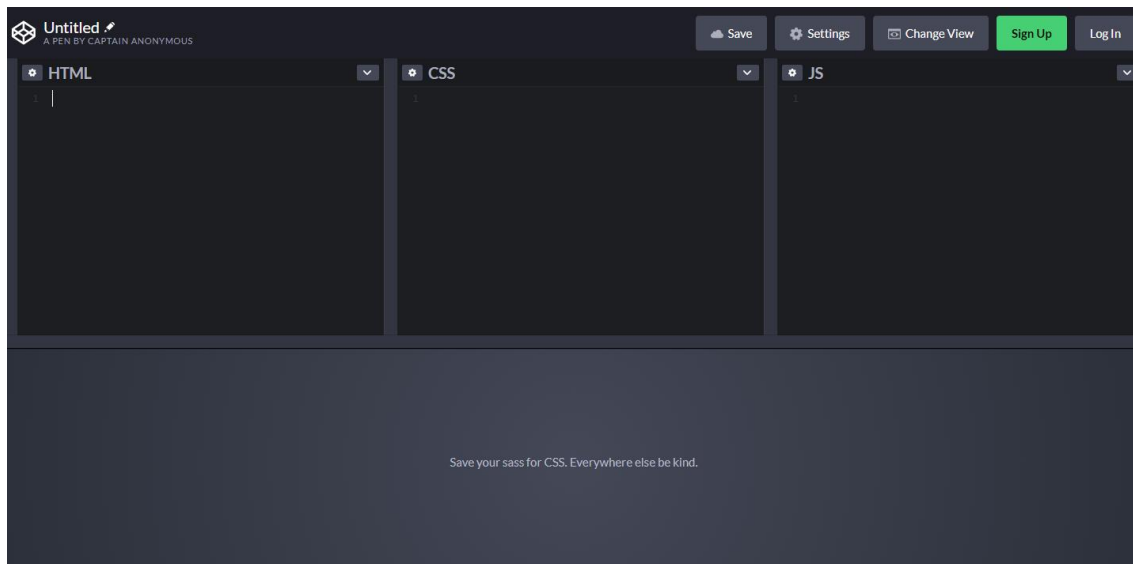
**Sololearn** se trata de un sitio web y aplicación para iOS y Android. Esta aplicación cuenta con una amplia comunidad de usuarios para el aprendizaje de lenguajes de programación (Java, C, C++, HTML, CSS, Javascript) y fundamentos de algoritmia, desde nivel principiante hasta nivel profesional.

Sitio web oficial: [www.sololearn.com](http://www.sololearn.com)



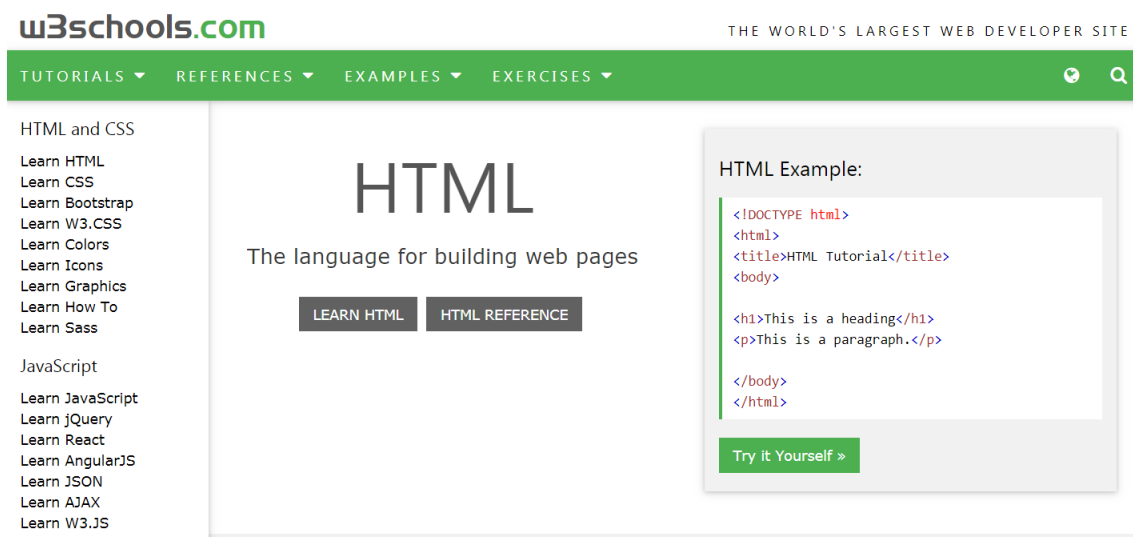
## 7. Codepen

**Consejo:** Si quieres comenzar a hacer pruebas con propiedades básicas de CSS, puedes utilizar [CodePen](https://codepen.io/), una plataforma web que te permite crear contenido HTML5, CSS3 y Javascript, prevvisualizando al vuelo el documento final, sin necesidad de recargar el documento.



## 8. W3schools

W3Schools es un sitio web para aprender tecnologías web en línea. Contiene tutoriales de HTML, CSS, JavaScript, SQL, PHP, XML y otras tecnologías. W3Schools presenta cientos de ejemplos de código.



## 9. Bibliografía

Los materiales para la creación de este manual han sido extraídos de la siguiente fuente:

- <https://lenguajecss.com/>. Autor: José Román Hernández (Profesor en la Oficina del Software Libre de la Universidad de La Laguna,)

