

Unidad 4. Creación de interfaces web utilizando estilos

Pseudoclases y pseudoelementos

Contenido

1. LAS PSEUDOCASES DINÁMICAS DE VÍNCULO.....	3
2. LAS PSEUDOCASES DINÁMICAS DE ACCIÓN.....	4
3. LA PSEUDOCASE DE DESTINO.....	5
4. LA PSEUDOCASE DE IDIOMA.....	6
5. LAS PSEUDOCASES DE ESTADO.....	7
6. LA PSEUDOCASE RAÍZ.....	10
7. LAS PSEUDOCASES DE PRIMER Y ÚLTIMO HIJO.....	10
8. LA PSEUDOCASE DE HIJOS EN LA POSICIÓN.....	11
9. LA PSEUDOCASE DE HIJOS EN LA POSICIÓN INVERSA.....	13
10. LAS PSEUDOCASES DEL PRIMER Y DEL ÚLTIMO HIJO DE UN TIPO.....	13
11. LAS PSEUDOCASES DE LOS PRIMEROS Y LOS ÚLTIMOS HIJOS DE UN TIPO.....	13
12. LA PSEUDOCASE DE LOS ELEMENTOS SIN HERMANOS.....	14
13. LA PSEUDOCASE DE LOS ELEMENTOS SIN HERMANOS DE UN TIPO.....	15
14. LA PSEUDOCASE DE LOS ELEMENTOS VACÍOS.....	16
15. LA PSEUDOCASE DE NEGACIÓN.....	17
16. LOS PSEUDOELEMENTOS.....	18
16.1. ::FIRST-LINE Y ::FIRST-LETTER.....	18
16.2. ::BEFORE Y ::AFTER.....	19
16.3. ::SELECTION.....	19
16.4. CONTENIDO GENERADO: CONTENT.....	20
16.4.1. Generar contadores: content, counter-increment y counter-reset.....	21

1. LAS PSEUDOCASES DINÁMICAS DE VÍNCULO

Las pseudoclases permiten aplicar formato a elementos que están fuera del árbol estructural del documento o a elementos que no pueden seleccionarse con los demás selectores.

Las pseudoclases dinámicas de vínculo se utilizan para dar formato a los vínculos. Podremos detectar diversos estados en los vínculos, relacionados con el uso que de ellos hacen o han hecho los visitantes:

- Estado del vínculo no visitado (:link).
- Estado del vínculo visitado, es decir, aquel en el que el visitante ha hecho un clic (:visited).
- Estado del vínculo cuando el ratón pasa por encima (:hover).
- Estado del vínculo en el momento de hacer clic con el ratón (:active).

Código para este ejemplo (**02_05.html**):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases dinámicas de vínculo</title>
  <meta charset="UTF-8" />
  <style>
    a {
      text-decoration: none;
    }
    a:link {
      background-color: lightyellow;
    }
    a:visited {
      background-color: lightblue;
    }
    a:hover {
      background-color: lightgreen;
    }
    a:active {
      background-color: lightpink;
    }
  </style>
</head>
<body>
<p><a href="#">Mi vínculo</a></p>
</body>
</html>
```

Estos son los diferentes formatos que aparecen en función del uso del vínculo:

- El vínculo (<a>) aparecerá con un fondo amarillo claro (lightyellow) cuando aún no se haya usado (a:link).
- El vínculo (<a>) aparecerá con un fondo verde claro (lightgreen) cuando el ratón pase por encima de él (a:hover).

- El vínculo (<a>) aparecerá con un fondo rosa claro (lightpink) cuando se haga clic en él (a:active).
- El vínculo (<a>) aparecerá con un fondo azul claro (lightblue) cuando se haya usado (a:visited).

2. LAS PSEUDOCASES DINÁMICAS DE ACCIÓN

Las pseudoclases dinámicas de acción con frecuencia se utilizan en los formularios. Sirven para dar formato a un elemento del formulario cuando el visitante usa su ratón en dicho elemento. Volvemos a encontrar las pseudoclases anteriores :hover y :active. Y, además, dispondremos de :focus, que indica el uso del elemento; por ejemplo, cuando el visitante escriba un valor en un campo de texto.

He aquí un ejemplo de formulario (02_06.html):

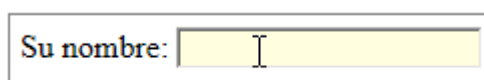
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases dinámicas de acción</title>
  <meta charset="UTF-8" />
  <style>
    input:hover {
      background-color: lightyellow;
    }
    input:focus {
      background-color: lightblue;
    }
  </style>
</head>
<body>
<form method="post" action="#">
  <label for="nombre">Su nombre: </label>
  <input type="text" name="nombre" value="" />
</form>
</body>
</html>
```

He aquí los diferentes formatos obtenidos en función del uso que haga el visitante:

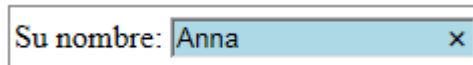
- El campo de texto del formulario (<input>) aparecerá con un fondo blanco cuando no se haya usado.

Una captura de pantalla de un formulario web. A la izquierda, el texto "Su nombre:" en color azul. A la derecha, un campo de entrada de texto rectangular con un fondo blanco y un borde gris.

- El campo de texto del formulario (<input>) aparecerá con un fondo amarillo claro (lightyellow) cuando el ratón pase por encima.

Una captura de pantalla de un formulario web. A la izquierda, el texto "Su nombre:" en color azul. A la derecha, un campo de entrada de texto rectangular con un fondo amarillo claro y un borde gris. El cursor del ratón está sobre el campo.

- El campo de texto del formulario (<input>) aparecerá con un fondo azul claro (lightblue) cuando se haya utilizado.



3. LA PSEUDOCCLASE DE DESTINO

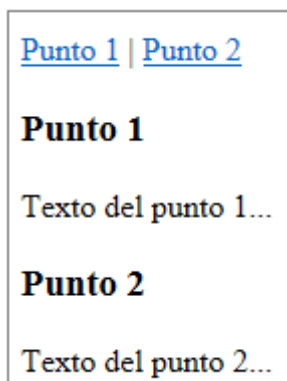
La pseudoclase de destino :target permite modificar el aspecto de un elemento en función del uso de un vínculo (<a>). Tomemos un ejemplo concreto: disponemos de un documento con elementos (<section>) identificados (id) y vínculos (<a>) que señalan a esos elementos identificados.

Veamos el código de este ejemplo (**02_07.html**):

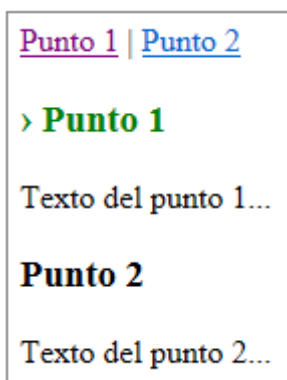
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>La pseudoclase de vínculo</title>
  <meta charset="UTF-8" />
  <link href="#" rel="stylesheet" />
  <style>
    section:target h3 {
      color: green;
    }
    section:target h3::before {
      content: "\203A\00a0" ;
    }
  </style>
</head>
<body>
<nav>
  <p><a href="#punto1">Punto 1</a> | <a href="#punto2">Punto 2</a></p>
</nav>
<section id="punto1">
  <h3>Punto 1</h3>
  <p>Texto del punto 1...</p>
</section>
<section id="punto2">
  <h3>Punto 2</h3>
  <p>Texto del punto 2...</p>
</section>
</body>
</html>
```

En el caso de este formato específico, apuntamos al elemento <h3> contenido en el elemento section: section:target<h3>. Le aplicamos un color de texto verde (green). Además, añadimos delante (:before) el elemento <h3> de los caracteres (content): el símbolo mayor que (\203A) y un espacio de no separación (\00a0).

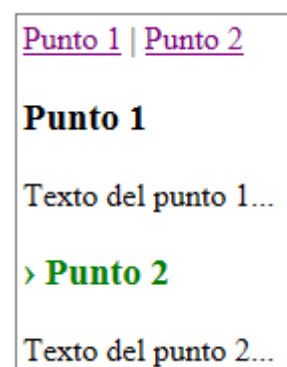
Cuando se carga la página, obtenemos este formato:



Cuando el visitante hace clic en el vínculo **Punto 1**, el texto del h3 «**Punto 1**» se muestra en verde, con el siguiente signo delante de los caracteres: «>».



Lo mismo sucede cuando el visitante hace clic en el vínculo **Punto 2**.



4. LA PSEUDOCASE DE IDIOMA

La pseudoclase `:lang` permite establecer un formato específico según el idioma definido con el atributo `lang`.

He aquí un ejemplo sencillo (**02_08.html**):

```
<!DOCTYPE html>  
<html lang="es">
```

```
<head>
  <title>La pseudoclase de idioma</title>
  <meta charset="UTF-8" />
  <style>
    span:lang(es) {background-color:lightblue}
    span:lang(en) {background-color:lightgreen}
    span:lang(it) {background-color:lightpink}
  </style>
</head>
<body>
<p lang="es">Texto <spanlang="es">en español</span>. Ahora
<span lang="en">in english</span> y para finalizar<span
lang="it">in italiano</span>.</p>
</body>
</html>
```

Este es el resultado:

Texto en español. Ahora in english y para finalizar in italiano.

5. LAS PSEUDOCASES DE ESTADO

Las pseudoclases de estado resultan muy útiles con los objetos de formulario: pueden resaltar los objetos activos, desactivados y marcados. De este modo, el usuario sabe lo que ha hecho y lo que está activo o inactivo.

Estas nuevas pseudoclases son:

- `:enabled`: para el estado activo del objeto de formulario.
- `:disabled`: para el estado desactivado del objeto de formulario.
- `:checked`: para el estado marcado.

Veamos el código HTML/CSS de este ejemplo (**02_09.html**):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases de estado</title>
  <meta charset="UTF-8" />
  <style>
    #acuerdo:checked+label {
      background-color: gold;
      font-weight: bold;
    }
    #hombre:checked+label, #mujer:checked+label {
      font-style: italic;
    }
    :enabled {
      background-color: lightgreen;
    }
    :disabled {
      background-color: lightcoral;
    }
  </style>
```

```
</head>
<body>
<form id="form1" method="#" action="#">
<p>
  <labelfor="nombre">Su nombre: </label>
  <input type="text" id="nombre" />
</p>
<p>
  <labelfor="edad">Su edad: </label>
  <input type="text" id="edad" disabled="disabled"/>
</p>
<p>
  <input type="radio" name="sexo" id="hombre" value="hombre" />
  <labelfor="hombre">Hombre</label><br/>
  <input type="radio" name="sexo" id="mujer" value="mujer" />
  <labelfor="hombre">Mujer</label>
</p>
<p>
  <input type="checkbox" id="acuerdo" />
  <labelfor="acuerdo">Estoy de acuerdo con las condiciones de uso</label>
</p>
</form>
</body>
</html>
```

Este formulario muestra dos campos de introducción de datos; el primero, para escribir el nombre, puede usarse (enabled), mientras que el segundo, para escribir la edad, no permite la escritura (disabled).

A continuación disponemos de dos botones de opción para elegir el sexo (Hombre o Mujer).

Finalmente, disponemos de una casilla de verificación.

El objetivo es resaltar los objetos de formulario en función de su estado: utilizable, no utilizable y marcado.

En los estilos CSS, el primer selector apunta a la etiqueta (label) de la casilla de verificación (#acuerdo) si está marcada (checked). En efecto, queremos resaltar la etiqueta de la casilla de verificación, y no la casilla propiamente dicha; por eso usamos un selector adyacente.

El segundo selector apunta al botón de opción marcado.

Los otros dos estilos simplemente permiten mostrar un color de fondo según el estado del campo.

Este es el aspecto que ofrece el formulario cuando el usuario aún no ha efectuado ninguna acción:

Su nombre:

Su edad:

☐ Hombre

☐ Mujer

☐ Estoy de acuerdo con las condiciones de uso

El usuario puede escribir en el primer campo activo (fondo verde), pero no en el segundo, ya que está inactivo (fondo rojo):

Su nombre:

Su edad:

☐ Hombre

☐ Mujer

☐ Estoy de acuerdo con las condiciones de uso

Si el usuario marca un botón de opción, la etiqueta pasa a mostrarse en cursiva:

Su nombre:

Su edad:

☐ Hombre

☒ *Mujer*

☐ Estoy de acuerdo con las condiciones de uso

El usuario ha marcado la casilla de verificación; la etiqueta aparece en negrita sobre un fondo dorado:

Su nombre:

Su edad:

☐ Hombre

☒ *Mujer*

☒ **Estoy de acuerdo con las condiciones de uso**

Existe otro estado para los botones de opción y las casillas de verificación. Se trata del estado indeterminado: `:indeterminate`. Este indica que el objeto no está ni marcado ni no marcado.

Aquí tienes otro ejemplo interesante del uso de `:checked` (ejemplo-checked.htm).

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Aprende a decir :not() - campusMVP</title>
  <link href="//fonts.googleapis.com/css?family=Open+Sans:400,500,600,700"
rel="stylesheet" type="text/css">
  <style>
    body {
      font-family: 'Open Sans', 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      font-size:1rem;
    }
    .opciones {
      padding:5px;
    }
    .otra-opcion div {
      margin:5px 5px 5px 26px;
    }
    .otra-opcion div{
      display: none;
    }
    .otra-opcion input:checked + label + div{
      display: block;
    }
  </style>
</head>
<body>
  <form>
    <div class="opciones opcion1">
      <input class="opciones" id="opcion1" type="radio" name="opcion"
value="Opción 1">
      <label for="opcion1">Opción 1</label>
    </div>
    <div class="opciones opcion2">
      <input class="opciones" id="opcion2" type="radio" name="opcion"
value="Opción 2">
      <label for="opcion1">Opción 2</label>
    </div>
    <div class="opciones otra-opcion">
      <input id="otra-opcion" type="radio" name="opcion" value="Otra opción">
      <label for="otra-opcion">Otra</label>
      <div>
        <label for="especifica">Por favor, especifica:</label>
        <input id="especifica" name="especifica" type="text">
      </div>
    </div>
  </form>
</body>
</html>
```

6. LA PSEUDOCLEASE RAÍZ

Las CSS ofrecen una serie de pseudoclases de estructura que permiten apuntar de forma precisa a los elementos estructurales de las páginas web.

La pseudoclase `:root` permite apuntar a la raíz de la página web, es decir, el elemento `<html>`. La diferencia con respecto al selector de elemento HTML es simple: `:root` es jerárquicamente superior.

7. LAS PSEUDOCLEASES DE PRIMER Y ÚLTIMO HIJO

La pseudoclase `:first-child` permite apuntar al primer hijo de un elemento padre.

He aquí un ejemplo sencillo (**02_10.html**):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases de primer y último hijo</title>
  <meta charset="UTF-8" />
  <style>
    li:first-child {
      font-weight: bold;
      color: green;
    }
  </style>
</head>
<body>
<ul>
  <li>Manzana</li>
  <li>Plátano</li>
  <li>Fresa</li>
</ul>
</body>
</html>
```

Cuando se visualice este código, el primer ítem `` se mostrará en verde y negrita.

- 
- **Manzana**
 - Plátano
 - Fresa

Siguiendo exactamente el mismo principio, la pseudoclase `:last-child` permite apuntar al último hijo de un elemento padre.

8. LA PSEUDOCLEASE DE HIJOS EN LA POSICIÓN

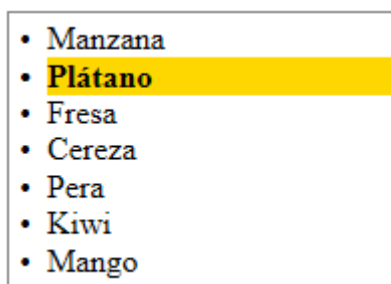
La pseudoclase `:nth-child(x)` permite apuntar a los hijos en la posición `x` de un elemento padre. La posición del hijo (o hijos) viene dado por el argumento `x` entre paréntesis.

El valor puede ser un número fijo: 2 o 5, por ejemplo.

Ejemplo (**02_11.html**):

```
<!DOCTYPEhtml>
<htmllang="es">
<head>
  <title>La pseudoclase de primeros hijos</title>
  <meta charset="UTF-8" />
  <style>
    li:nth-child(2) {
      font-weight: bold;
      background-color: gold;
    }
  </style>
</head>
<body>
<ul>
  <li>Manzana</li>
  <li>Plátano</li>
  <li>Fresa</li>
  <li>Cereza</li>
  <li>Pera</li>
  <li>Kiwi</li>
  <li>Mango</li>
</ul>
</body>
</html>
```

Resultado que se muestra:



El valor puede referirse a los hijos pares (even) o a los impares (odd).

Veamos el ejemplo anterior (**02_12.html**) con `li:nth-child(odd)`.

- **Manzana**
- Plátano
- **Fresa**
- Cereza
- **Pera**
- Kiwi
- **Mango**

El valor x también puede ser un cálculo en formato: $an+b$. n representa un valor que comienza por 0 y que se incrementa en 1 con cada paso, y puede ser positivo o negativo.

Primer ejemplo (**02_13.html**):

`:nth-child(n+3)` permite dejar al margen los dos primeros hijos.

Este es el cálculo:

$0+3=3$, el tercer hijo,

$1+3=4$, el cuarto hijo,

$2+3=5$, el quinto hijo...

Esto es lo que se muestra:

- Manzana
- Plátano
- **Fresa**
- **Cereza**
- **Pera**
- **Kiwi**
- **Mango**

Segundo ejemplo (**02_14.html**):

`:nth-child(3n+1)` permite apuntar a un hijo de cada 3 comenzando por el primero.

Este es el cálculo:

$(3 \times 0)+1=1$, el primer hijo,

$(3 \times 1)+1=4$, el cuartohijo,

$(3 \times 2)+1=7$, el séptimo hijo,

$(3 \times 3)+1=10$, el décimo hijo...

Esto es lo que se muestra:

- **Manzana**
- Plátano
- Fresa
- **Cereza**
- Pera
- Kiwi
- **Mango**

9. LA PSEUDOCLEASE DE HIJOS EN LA POSICIÓN INVERSA

La pseudoclase `:nth-last-child()` permite apuntar a los últimos hijos de un elemento padre en la posición *x*. Funciona del mismo modo que `:nth-child()` pero empezando desde el último elemento hijo.

10. LAS PSEUDOCLEASES DEL PRIMER Y DEL ÚLTIMO HIJO DE UN TIPO

Las nuevas pseudoclases `:first-of-type` y `:last-of-type` permiten apuntar al primer y al último elemento de un tipo especificado.

He aquí un ejemplo (**02_15.html**) de la lista `` anterior, tomando como elementos de destino el primer y el último ítem ``.

```
li:first-of-type, li:last-of-type {  
    background-color: gold;  
}
```

Se muestra lo siguiente:

- **Manzana**
- Plátano
- Fresa
- Cereza
- Pera
- Kiwi
- **Mango**

11. LAS PSEUDOCLEASES DE LOS PRIMEROS Y LOS ÚLTIMOS HIJOS DE UN TIPO

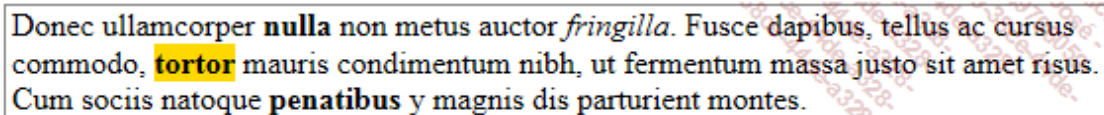
La pseudoclase `:nth-of-type()` permite apuntar al *n*ésimo elemento de un tipo especificado. Los argumentos posibles son idénticos a los que hemos visto con la pseudoclase `:nth-child()`.

He aquí un ejemplo (**02_16.html**) muy sencillo, con la aplicación de formato (y) a determinadas palabras en un párrafo (<p>):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases</title>
  <meta charset="UTF-8" />
  <style>
    strong:nth-of-type(2) {
      background-color: gold;
    }
  </style>
</head>
<body>
<p>Donecullamcorper<strong>nulla</strong>      non      metus auctor<em>fringilla</em>.
Fuscedapibus,                                tellus                                ac
cursus commodo,<strong>tortor</strong>mauriscondimentumnibh,utfermentummassajusto
sit ametrisus. Cum sociisnatoque<strong>penatibus</strong> y magnisdisparturient
montes.</p>
</body>
</html>
```

El elemento <p> tiene como primer elemento hijo un , como segundo un , como tercero y cuarto otro . Deseamos apuntar al segundo , y no al segundo elemento hijo.

Esto es lo que se muestra:



Donec ullamcorper **nulla** non metus auctor *fringilla*. Fusce dapibus, tellus ac cursus
commodo, **tortor** mauris condimentum nibh, ut fermentum massa justo sit amet risus.
Cum sociis natoque **penatibus** y magnis dis parturient montes.

La clase :nth-last-of-type() permite apuntar al enésimo último elemento de un tipo dado.

12. LA PSEUDOCLEASE DE LOS ELEMENTOS SIN HERMANOS

La pseudoclase :only-child apunta a los elementos que no tienen hermanos.

Tomemos el ejemplo de un texto (<p>) en el que resaltamos algunos términos con . Deseamos destacar los párrafos que incluyen un único resalte .

Este es el código de este ejemplo (**02_17.html**).

El primer párrafo <p> incluye dos ; el segundo, uno solo.

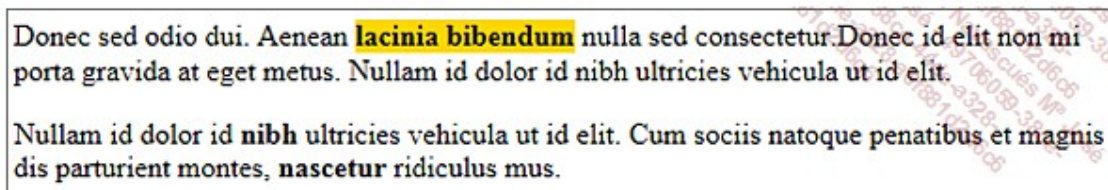
```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>La pseudoclase de los elementos sin hermanos</title>
  <meta charset="UTF-8" />
  <style>
    strong:only-child {
```

```

        background-color: gold;
    }
</style>
</head>
<body>
<p>Donecsedodio                                dui.
Aenean<strong>laciniabibendum</strong>nullasedconsectetur.Donec id elit non mi
portagravida at egetmetus. Nullam id dolor id nibh ultriciesvehiculaut id
elit.</p>
<p>Nullam id dolor id <strong>nibh</strong>ultricies vehiculaut id elit. Cum
sociisnatoquepenatibus etmagnis dis parturient montes,
<strong>nascetur</strong>ridiculus mus.</p>
</body>
</html>

```

Esto es lo que se muestra:



13. LA PSEUDOCLEASE DE LOS ELEMENTOS SIN HERMANOS DE UN TIPO

La pseudoclase `:only-of-type` apunta a los elementos de un tipo dado que no tienen hermanos.

Retomemos en este ejemplo (**02_18.html**) el principio precedente añadiendo un resalte `` en el primer párrafo.

```

<p>Donecsedodio                                dui.
Aenean<strong>laciniabibendum</strong>nullasedconsectetur.Donec id elit non mi
portagravida at egetmetus. Nullam id dolor id
<em>nibhultricies</em>vehiculaut id elit.</p>
<p>Nullam id dolor id <strong>nibh</strong>ultricies vehiculaut id elit. Cum
sociisnatoquepenatibuset magnis dis parturient montes, <strong>nascetur</strong>
ridiculus mus.</p>

```

El estilo modificado:

```

strong:only-of-type {
    background-color: gold;
}

```

Con el estilo CSS anterior, el `` del primer párrafo ya no aparece con un fondo de color, ya que este párrafo contiene ahora dos resaltes. El `` no está solo; también hay un ``. Con la pseudoclase `:only-of-type` ahora podemos especificar el tipo de elemento de forma aislada.

Esto es lo que se muestra:

Donec sed odio dui. Aenean **lacinia bibendum** nulla sed consectetur. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id nibh ultricies vehicula ut id elit.

Nullam id dolor id nibh ultricies vehicula ut id elit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

14. LA PSEUDOCLEASE DE LOS ELEMENTOS VACÍOS

La pseudoclase `:empty` apunta a elementos que están vacíos, que no contienen ningún elemento. Esto puede resultar práctico si trabaja en una página web que se construye dinámicamente; puede incluir elementos (``, `<td>`, ``...) que están vacíos. Con esta pseudoclase es posible resaltarlas.

Veamos el código de este ejemplo (**02_19.html**):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>La pseudoclase de los elementos vacíos</title>
  <meta charset="UTF-8" />
  <style>
    li:empty {
      background-color: gold;
    }
  </style>
</head>
<body>
<ul>
  <li>Manzana</li>
  <li>Plátano</li>
  <li></li>
  <li>Cereza</li>
  <li></li>
  <li>Kiwi</li>
  <li>Mango</li>
</ul>
</body>
</html>
```

Esto es lo que se muestra:

- Manzana
- Plátano
-
- Cereza
-
- Kiwi
- Mango

15. LA PSEUDOCLEASE DE NEGACIÓN

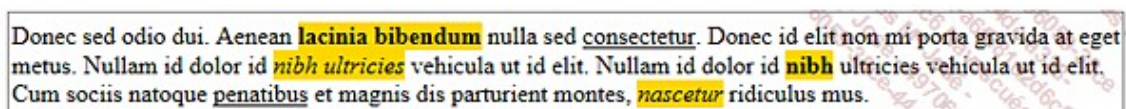
La pseudoclase de negación `:not` permite apuntar a todos los elementos HTML que no son el especificado.

Veamos el código de este ejemplo (**02_20.html**):

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Las pseudoclases</title>
  <meta charset="UTF-8" />
  <style>
    p:not(u) {
      background-color: gold;
    }
  </style>
</head>
<body>
<p>Donecsedodio                                dui.
Aenean<strong>lacinia bibendum</strong>nullased<u>consectetur</u>. Donec id elit
non mi portagravida at egetmetus. Nullam id dolor id
<em>nibhultricies</em>vehiculaut id elit. Nullam id dolor id
<strong>nibh</strong>ultriciesvehiculaut id elit. Cum
sociisnatoque<u>penatibus</u> et magnis dis parturient montes,
<em>nascetur</em>ridiculus mus.</p>
</body>
</html>
```

En el párrafo `<p>`, hay tres tipos de formato: negrita (``), cursiva (``) y subrayado (`<u>`). El selector va a apuntar, en el párrafo `<p>`, a todos los elementos hijos que no están subrayados `<u>`: `p :not(u)`. Todos estos elementos aparecerán con un fondo dorado.

Esto es lo que se muestra:



Donec sed odio dui. Aenean **lacinia bibendum** nulla sed consectetur. Donec id elit non mi porta gravida at eget metus. Nullam id dolor id **nibh** *ultricies* vehicula ut id elit. Nullam id dolor id **nibh** *ultricies* vehicula ut id elit. Cum sociis natoque penatibus et magnis dis parturient montes, **nascetur** ridiculus mus.

Todos los elementos hijos tienen un fondo dorado, excepto los subrayados.

16. LOS PSEUDOELEMENTOS

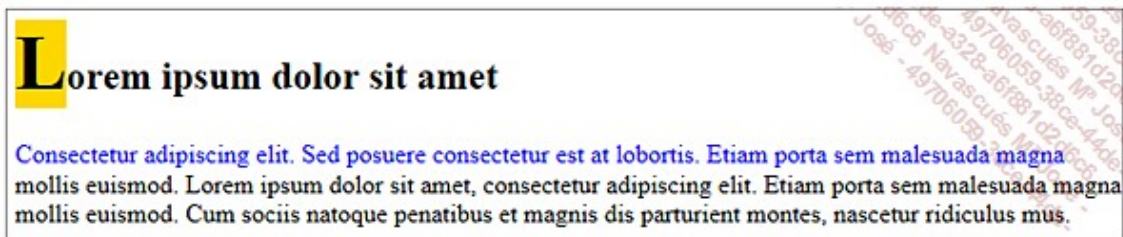
Los pseudoelementos aplican un formato particular a la tipografía del texto: en la primera línea y sobre la primera letra de una palabra.

16.1. ::FIRST-LINE Y ::FIRST-LETTER

He aquí un ejemplo de su uso (**02_21.html**): la primera letra del título <h2> se mostrará con un tamaño el doble de grande y sobre un fondo dorado; la primera línea del párrafo <p> aparecerá en azul.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Los pseudoelementos de texto</title>
  <meta charset="UTF-8" />
  <style>
    h2::first-letter {
      font-size: 2em;
      background-color: gold;
    }
    p::first-line {
      color: blue;
    }
  </style>
</head>
<body>
<h2>Lorem ipsum dolor sit amet</h2>
<p>Consectetur adipiscing elit...</p>
</body>
</html>
```

Esto es lo que se muestra:



Nota: El pseudoelemento ::first-letter solo se aplica a elementos de tipo bloque.

Al pseudoelemento ::first-letter se le pueden aplicar las siguientes propiedades:

- Propiedades de las fuentes
- Propiedades del color
- Propiedades del fondo
- Propiedades del margen
- Propiedades del relleno
- Propiedades del borde

- text-decoration
- vertical-align (solo si "float" es "none")
- text-transform
- line-height
- float
- clear

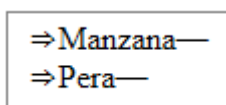
16.2. ::BEFORE Y ::AFTER

Los dos pseudoelementos, **::before** y **::after**, añaden contenido antes o después de los elementos apuntados. El contenido generado puede ser texto o imágenes.

He aquí un ejemplo (**02_22.html**) sencillo de uso de este contenido generado durante la creación de una lista. Suprimimos las viñetas estándares de la lista `` para añadir una flecha delante y una raya detrás.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <title>Los pseudoelementos de contenido</title>
  <meta charset="UTF-8" />
  <style>
    li {
      list-style: none;
    }
    li::before {
      content: "\21d2";
    }
    li::after {
      content: "\2014";
    }
  </style>
</head>
<body>
<ul>
  <li>Manzana</li>
  <li>Pera</li>
</ul>
</body>
</html>
```

Esto es lo que se muestra:



```
⇒Manzana—
⇒Pera—
```

16.3. ::SELECTION

El pseudoelemento **::selection** permite formatear el texto seleccionado en un sitio web.

El siguiente ejemplo hace que el texto seleccionado aparezca de color rojo sobre fondo Amarillo.

```
::selection {
color: red;
background: yellow;
}
```

Funciona solo en algunos navegadores.

16.4. CONTENIDO GENERADO: CONTENT

Los pseudo-elementos `:before` y `:after` permiten añadir contenido a un elemento desde la hoja de estilo, al principio o al final del elemento.

Nota: En Firefox y Google Chrome, el contenido generado no puede seleccionarse con el ratón, pero en Internet Explorer sí.

El contenido generado mediante la propiedad `content` puede incluir texto:

```
p.cuidado:before{
content:"Aviso: ";
font-weight: bold;
text-decoration:underline;
}
```

Este párrafo es un párrafo `<p>` sin clase.

Aviso: Este párrafo es un párrafo `<p>` con clase "cuidado".

Este párrafo es un párrafo `<p>` sin clase.

```
p.autor-barto:after{
content:"          (escrito
Barto).";
}
```

Este párrafo es un párrafo `<p>` sin clase.

Este párrafo es un párrafo `<p>` con clase "autor-barto".
(escrito por Barto).

Este párrafo es un párrafo `<p>` sin clase.

El contenido generado mediante la propiedad `content` puede incluir una imagen indicando su URI:

```
p.ff:before{
content:url("../varios/iconos/icono_ff.svg");
}
```

Este párrafo es un párrafo `<p>` sin clase.

Este párrafo es un párrafo `<p>` con clase "ff".

Este párrafo es un párrafo `<p>` sin clase.

No es válido incluir etiquetas html en la propiedad `content`, es decir que no sólo los navegadores no hacen caso y lo incluyen como texto, no como código html, sino que al validar la hoja de estilo daría error:

```
p.cuidado2:before{
content:"<em>Aviso:</em>";
}
```

Este párrafo es un párrafo `<p>` sin clase.

`Aviso:` Este párrafo es un párrafo `<p>` con clase

```

    "cuidado2".
}

```

Este párrafo es un párrafo <p> sin clase.

Los pseudo-elementos `:before` y `:after` se utilizan en combinación con la propiedad `content` de CSS para añadir contenidos antes o después del contenido original de un elemento.

Las siguientes reglas CSS añaden el texto Capítulo - delante de cada título de sección <h1> y el carácter . detrás de cada párrafo de la página:

```
h1:before { content: "Capítulo - "; }
```



```
p:after { content: "."; }
```

El contenido insertado mediante los pseudo-elementos `:before` y `:after` se tiene en cuenta en los otros pseudo-elementos `:first-line` y `:first-letter`.


16.4.1.GENERAR CONTADORES: CONTENT, COUNTER-INCREMENT Y COUNTER-RESET

Se pueden generar contadores en los pseudo-elementos `:after` y `:before` mediante la propiedad `content` y el valor `counter(nombre_de_contador)`. El contador debe ponerse a cero con la propiedad `counter-reset` y aumentarse con la propiedad `counter-increment`.


En los ejemplos siguientes se ha definido un contador que se llama cuenta-parrafos. Este contador se genera al principio de cada párrafo, se pone a cero con el elemento <pre> y se incrementa en cada párrafo.

 <pre> pre { counter-reset: cuenta-parrafos; } p:before { content: counter(cuenta-parrafos); counter-increment: cuenta-parrafos; } </pre>	<p>Ejemplo de contadores (este párrafo es un <pre>)</p> <p>1Este párrafo es un párrafo <p> sin clase.</p> <p>2Este párrafo es un párrafo <p> sin clase.</p> <p>3Este párrafo es un párrafo <p> sin clase.</p> <p>4Este párrafo es un párrafo <p> sin clase.</p> <p>5Este párrafo es un párrafo <p> sin clase.</p>
 <pre> pre { counter-reset: cuenta-parrafos; } p:before { content: counter(cuenta-parrafos); counter-increment: cuenta-parrafos; } </pre>	<p>Ejemplo de contadores (este párrafo es un <pre>)</p> <p>1Este párrafo es un párrafo <p> sin clase.</p> <p>2Este párrafo es un párrafo <p> sin clase.</p> <p>3Este párrafo es un párrafo <p> sin clase.</p> <p>Ejemplo de contadores (este párrafo es un <pre>)</p> <p>1Este párrafo es un párrafo <p> sin clase.</p> <p>2Este párrafo es un párrafo <p> sin clase.</p>


El contador puede incrementarse en cualquier cantidad entera, como muestra el ejemplo siguiente:

 <pre>pre { counter-reset: cuenta-parrafos; } p:before { content: counter(cuenta-parrafos); counter-increment: cuenta-parrafos 10; }</pre>	<p>Ejemplo de contadores (este párrafo es un <pre>)</p> <p>10Este párrafo es un párrafo <p> sin clase.</p> <p>20Este párrafo es un párrafo <p> sin clase.</p> <p>30Este párrafo es un párrafo <p> sin clase.</p> <p>40Este párrafo es un párrafo <p> sin clase.</p> <p>50Este párrafo es un párrafo s<p> in clase.</p>
--	--


Junto con el contador se puede generar texto, como muestra el ejemplo siguiente:

 <pre>pre { counter-reset: cuenta-parrafos; } p:before { content: counter(cuenta-parrafos) ". "; counter-increment: cuenta-parrafos; }</pre>	<p>Ejemplo de contadores (este párrafo es un <pre>)</p> <ol style="list-style-type: none"> 1. Este párrafo es un párrafo <p> normal y corriente. 2. Este párrafo es un párrafo <p> normal y corriente. 3. Este párrafo es un párrafo <p> normal y corriente. <p>Ejemplo de contadores (este párrafo es un <pre>)</p> <ol style="list-style-type: none"> 1. Este párrafo es un párrafo <p> normal y corriente. 2. Este párrafo es un párrafo <p> normal y corriente.
--	--

El contador predeterminado muestra números enteros, pero se puede utilizar cualquiera de los estilos de listas como estilo de contador. El estilo se indica en la propiedad **content** mediante el valor **counter(nombre_de_contador, estilo_de_lista)**, como muestra el ejemplo siguiente:

 <pre>pre { counter-reset: cuenta-parrafos; } p:before { content: counter(cuenta-parrafos, upper-roman) ". "; counter-increment: cuenta-parrafos; }</pre>	<p>Ejemplo de contadores (este párrafo es un <pre>)</p> <ol style="list-style-type: none"> I. Este párrafo es un párrafo <p> normal y corriente. II. Este párrafo es un párrafo <p> normal y corriente. III. Este párrafo es un párrafo <p> normal y corriente. <p>Ejemplo de contadores (este párrafo es un <pre>)</p> <ol style="list-style-type: none"> I. Este párrafo es un párrafo <p> normal y corriente. II. Este párrafo es un párrafo <p> normal y corriente.
---	--

Se pueden utilizar varios contadores simultáneamente, como muestra el ejemplo siguiente:

 <pre>h4 { counter-reset: cuenta-parrafos; counter-reset: cuenta-aptados; } h5 { counter-reset: cuenta-parrafos; } h5:before { content: counter(cuenta-aptados, upper-alpha) ". "; counter-increment: cuenta-aptados; } p:before { content: counter(cuenta-aptados, upper-alpha) "-" counter(cuenta-parrafos) ". "; counter-increment: cuenta-parrafos; }</pre>	Ejemplo de contadores A. Apartado 1 A-1. Este párrafo es un párrafo <p> normal y corriente. A-2. Este párrafo es un párrafo <p> normal y corriente. A-3. Este párrafo es un párrafo <p> normal y corriente. B. Apartado 2 B-1. Este párrafo es un párrafo <p> normal y corriente. B-2. Este párrafo es un párrafo <p> normal y corriente.
---	---

Aquí tienes algunos ejemplos más de los pseudoelementos :before y :after.

<http://cssdemos.tupence.co.uk/before.htm>