# HTML5 Y CSS3

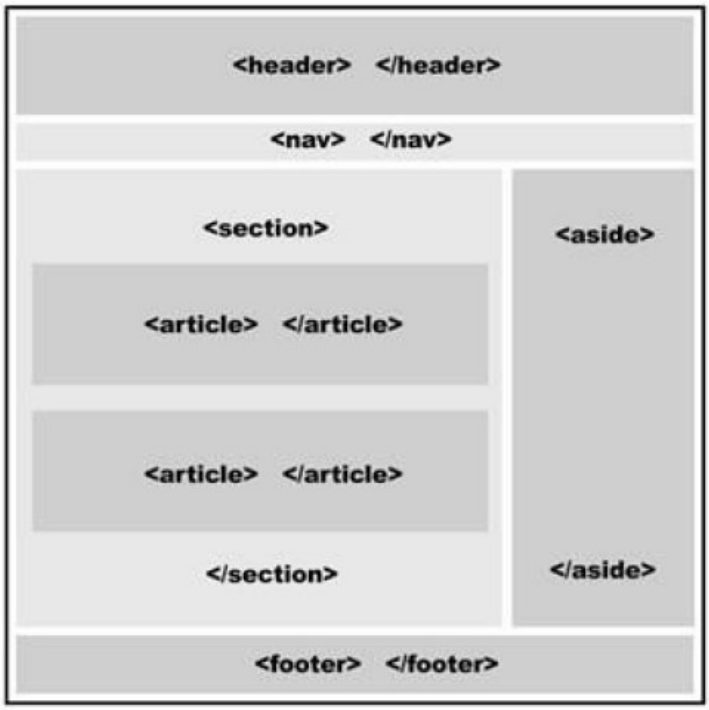
1. ETIQUETAS SEMÁNTICAS HTML5
2. CSS3
   - Selectores básicos
   - Selectores complejos
     - Selectores hijos y hermanos
     - Metodología BEM
3. MODELO DE CAJAS CON HTML Y CSS3
4. NUEVAS ETIQUETAS CSS3.
   - PREFIJOS CSS DE NAVEGADORES
5. LAS PSEUDOCLASES
6. VÍDEO Y AUDIO EN HTML
7. NUEVAS ETIQUETAS PARA FORMULARIOS
8. CSS AVANZADO

   ocultar texto u objetos(leer más)

   Rollovers y Sprites

## CAPITULO 1. ETIQUETAS SEMEMÁNTICAS HTML5

**HTML5 incorpora nuevos elementos que ayudan a identificar cada secci**ón del documento y organizar el cuerpo del mismo. En HTML5 las secciones más importantes son diferenciadas. A partir de ahora podemos decir al navegador para qué es cada sección.




*1-4. Representación visual de las etiquetas <article> que fueron incluidas para contener información relevante de la página web.*

Las etiquetas semánticas de HTML5 son:

**<header>** Este elemento presenta información introductoria y puede ser aplicado en diferentes secciones del
documento. Tiene el propósito de contener la cabecera de una sección pero también puede ser utilizado para
agrupar índices, formularios de búsqueda, logos, etc…

**<nav>** Este elemento indica una sección de enlaces con propósitos de navegación, como menús o índices. No todos los enlaces dentro de una página web tienen que estar dentro de un elemento **<nav>**, solo aquellos que forman partes de bloques de navegación.

**<section>** Este elemento representa una sección general del documento. Es usualmente utilizado para construir varios bloques de contenido (por ejemplo, columnas) con el propósito de ordenar el contenido que comparte una característica específica, como capítulos o páginas de un libro, grupo de noticias, artículos, etc…

**<aside>** Este elemento representa contenido que está relacionado con el contenido principal pero no es parte del mismo. Ejemplos pueden ser citas, información en barras laterales, publicidad, etc…

**<footer>** Este elemento representa información adicional sobre su elemento padre. Por ejemplo, un elemento **<footer>** insertado al final del cuerpo proveerá información adicional sobre el cuerpo del documento, como el pie normal de una página web. Puede ser usado no solo para el cuerpo sino también para diferentes secciones dentro del cuerpo, otorgando información adicional sobre estas secciones específicas.

**Y algunas etiquetas nuevas de HTML5 SON:**
**<article>** Este elemento representa una porción independiente de información relevante (por ejemplo, cada artículo de un periódico o cada entrada de un blog). El elemento **<article>** puede ser anidado y usado para mostrar una lista dentro de otra lista de ítems relacionados, como comentarios de usuarios en entradas de blogs, por ejemplo.

**<hgroup>** Este elemento es usado para agrupar elementos H cuando la cabecera tiene múltiples niveles (por ejemplo, una cabecera con título y subtítulo).

**<figure>** Este elemento representa una porción independiente de contenido (por ejemplo, imágenes, diagramas o videos) que son referenciadas desde el contenido principal. Esta es información que puede ser removida sin afectar el fluido del resto del contenido.

**<figcaption>** Este elemento es utilizado para mostrar una leyenda o pequeño texto relacionado con el contenido de un elemento **<figure>**, como la descripción de una imagen.

**<mark>** Este elemento resalta un texto que tiene relevancia en una situación en particular o que ha sido mostrado en respuesta de la actividad del usuario.

**<small>** Este elemento representa contenido al margen, como letra pequeña (por ejemplo, descargos, restricciones legales, declaración de derechos, etc… ).

**<cite>** Este elemento es usado para mostrar el título de un trabajo (libro, película, poema, etc… ).

**<address>** Este elemento encierra información de contacto para un elemento **<article>** o el documento completo. Es recomendable que sea insertado dentro de un elemento **<footer>**.

**<time>** Este elemento se utiliza para mostrar fecha y hora en formatos comprensibles por los usuarios y el navegador.
El valor para los usuarios es ubicado entre las etiquetas mientras que el específico para programas y navegadores es incluido como el valor del atributo **datetime**. Un segundo atributo optativo llamado **pubdate** es usado para indicar que el valor de **datetime** es la fecha de publicación.

## CAPÍTULO 2. HOJA DE ESTILOS EN CASCADA CSS

Existen diferentes formas de insertar reglas CSS en nuestra web.

- **Estilos en líneas-->(1)**
- **Estilos embebidos-->(2)**
- **Archivos externos-→(3)**

**Ejemplos**

```
<!DOCTYPE html>
<html lang="es">
        <head>
                <title>Este es el título del documento</title>
                <style>
                        p { font-size: 20px }
                </style>
                <link rel="stylesheet" href="misestilos.css">
        </head>
        <body>
                <p style="font-size: 20px">Mi texto</p>
        </body>
</html>
```

2

3

1

**No perder de vista esta otra posibilidad:**
Aunque generalmente se emplea la etiqueta <link> para enlazar los archivos CSS externos, también se puede utilizar la etiqueta <style>. La forma alternativa de incluir un archivo CSS externo se muestra a continuación:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de estilos CSS en un archivo externo</title>
```

```
<style type="text/css" media="screen">
  @import '/css/estilos.css';
</style>
</head>
```

En este caso, para incluir en la página HTML los estilos definidos en archivos CSS externos se utiliza una regla especial de tipo @import.

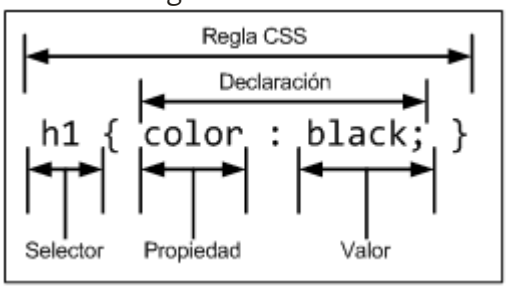
La URL del archivo CSS externo se indica mediante una cadena de texto encerrada con comillas simples o dobles o mediante la palabra reservada url(). De esta forma, las siguientes reglas @import son equivalentes:

```
@import '/css/estilos.css';
@import "/css/estilos.css";
```

```
@import url('/css/estilos.css');
@import url("/css/estilos.css");
```

**Definición de estilos**

La definición de estilos se lleva a cabo de la siguiente manera:




## Referencias

**Para referenciar los estilos se pueden usar diferentes técnicas**

- referencia por la palabra clave del elemento

```
p { font-size: 20px }
```

- referencia por el atributo **id**

```
#texto1 { font-size: 20px }
```

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
<link rel="stylesheet" href="misestilos.css">
</head>
<body>
<p id="texto1">Mi texto</p>
</body>
</html>
```

- referencia por el atributo **class**

```
.texto1 { font-size: 20px }
```

```
<html lang="es">
<head>
<title>Este texto es el título del documento</title>
```

```
<link rel="stylesheet" href="misestilos.css">
</head>
<body>
<p class="texto1">Mi texto</p>
<p class="texto1">Mi texto</p>
<p>Mi texto</p>
</body>
</html>
```

**referencia la clase llamada texto1** pero solo para los elementos de tipo **<p>.**

```
p.texto1 { font-size: 20px }
```

Entonces, **para saber cuándo y por qué utilizar ids o class,** es importante tener clara la definición de ambos.
El valor del atributo "id" de un elemento es único; es decir, no debería haber otro elemento con el mismo nombre de identificador (id) dentro de tu documento HTML.
Por otra parte, el valor del atributo "class", a diferencia del valor del atributo "id", puede ser utilizado en más de un elemento de tu documento HTML, esto nos es muy beneficioso cuando tenemos que aplicar los mismos estilos a diferentes elementos, dado que nos permite reducir las líneas de código en nuestro archivo .css, considerándose una buena práctica.

## Selectores básicos

**Selector descendiente:**
Selecciona los elementos que se encuentran dentro de otros elementos.

El selector del siguiente ejemplo selecciona todos los elementos  de la página que se encuentren dentro de un elemento <p>:

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

Si el código HTML de la página es el siguiente:

```
<p>
 ...
 <span>texto1</span>
 ...
 <a href="">...<span>texto2</span></a>
 ...
</p>
```

El selector p span selecciona tanto texto1 como texto2.

**Combinación de selectores**

CSS permite agrupar todas las reglas individuales en una sola regla con un selector múltiple. Para ello, se incluyen todos los selectores separados por una coma (,) y el resultado es que la siguiente regla CSS es equivalente a las tres reglas anteriores:

```
h1, h2, h3 {
  color: #8A8E27;
font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}
```

No debe confundirse el selector descendente con la combinación de selectores:

Ejemplo:

El estilo se aplica a todos los elementos "p", "a", "span" y "em"

```
p, a, span, em { text-decoration: underline; }
```

El estilo se aplica solo a los elementos "em" que se encuentran dentro de "p a span"

```
p a span em { text-decoration: underline; }
```

### Selectores avanzados

Utilizando solamente los selectores básicos es posible diseñar prácticamente cualquier página web. No obstante, CSS define otros selectores más avanzados que permiten simplificar las hojas de estilos. Desafortunadamente, el navegador Internet Explorer 6 y sus versiones anteriores no soportaban este tipo de selectores avanzados, por lo que su uso no era común hasta hace poco tiempo.

### Selector de hijos

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento. Se utiliza para seleccionar un elemento que es **hijo directo de otro elemento** y se indica mediante el "signo de mayor que" (>):

```
p > span { color: blue; }
<p><span>Texto1</span></p>
<p><a href="#"><span>Texto2</span></a></p>
```

En el ejemplo anterior, el selector p > span se interpreta como "cualquier elemento  que sea hijo directo de un elemento <p>", por lo que el primer elemento  cumple la condición del selector. Sin embargo, el segundo elemento  no la cumple porque es descendiente pero no es hijo directo de un elemento <p>.

El siguiente ejemplo muestra las diferencias entre el selector descendente y el selector de hijos:

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

<p><a href="#">Enlace1</a></p>
<p><span><a href="#">Enlace2</a></span></p>
```

El primer selector es de tipo descendente y por tanto se aplica a todos los elementos <a> que se encuentran dentro de elementos <p>. En este caso, los estilos de este selector se aplican a los dos enlaces.

Por otra parte, el selector de hijos obliga a que el elemento <a> sea hijo directo de un elemento <p>. Por lo tanto, los estilos del selector p > a no se aplican al segundo enlace del ejemplo anterior.

### Selector adyacente

El selector adyacente se emplea para seleccionar elementos que en el código HTML de la página se encuentran justo a continuación de otros elementos. Su sintaxis emplea el signo + para separar los dos elementos:

```
elemento1 + elemento2 { ... }
```

Si se considera el siguiente código HTML:

```
<body>
<h1>Titulo1</h1>

<h2>Subtítulo</h2>
...

<h2>Otro subtítulo</h2>
```

```
...
</body>
```

La página anterior dispone de dos elementos <h2>, pero sólo uno de ellos se encuentra inmediatamente después del elemento <h1>. Si se quiere aplicar diferentes colores en función de esta circunstancia, el selector adyacente es el más adecuado:

```
h2 { color: green; }
h1 + h2 { color: red }
```

Las reglas CSS anteriores hacen que todos los <h2> de la página se vean de color verde, salvo aquellos <h2> que se encuentran inmediatamente después de cualquier elemento <h1> y que se muestran de color rojo.

Técnicamente, los elementos que forman el selector adyacente deben cumplir las dos siguientes condiciones:

- ***elemento1*** **y** ***elemento2*** deben ser elementos hermanos, por lo que su elemento padre debe ser el mismo.
- ***elemento2*** debe aparecer inmediatamente después de ***elemento1*** en el código HTML de la página.

El siguiente ejemplo es muy útil para los textos que se muestran como libros:

```
p + p { text-indent: 1.5em; }
```

En muchos libros, suele ser habitual que la primera línea de todos los párrafos esté indentada, salvo la primera línea del primer párrafo. Con el selector ***p + p,*** se seleccionan todos los párrafos de la página que estén precedidos por otro párrafo, por lo que no se aplica al primer párrafo de la página.

## SELECTOR GENERAL DE ELEMENTOS HERMANOS

Otro de los nuevos selectores de CSS 3 es el "selector general de elementos hermanos". Su sintaxis es **`elemento1 ~ elemento2`** y s**elecciona el `elemento2` que es hermano de `elemento1` y se encuentra detrás en el código HTML.**

En el selector adyacente la condición adicional era que los dos elementos debían estar uno detrás de otro en el código HTML, mientras que ahora la única condición es que uno esté detrás de otro.

Si se considera el siguiente ejemplo:

```
h1 + h2 { ... }  /* selector adyacente */
h1 ~ h2 { ... }  /* selector general de hermanos */


<h1>...</h1>
<h2>...</h2>
<p>...</p>
<div>
  <h2>...</h2>
</div>
<h2>...</h2>
```

El primer selector (`h1 + h2`) sólo selecciona el primer elemento `<h2>` de la página, ya que es el único que cumple que es hermano de `<h1>` y se encuentra justo detrás en el código HTML.

Por su parte, el segundo selector (`h1 ~ h2`) selecciona todos los elementos `<h2>` de la página salvo el segundo. Aunque el segundo `<h2>` se encuentra detrás de `<h1>` en el código HTML, no son elementos hermanos porque no tienen el mismo elemento padre.

### Selector de atributos

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos HTML en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- ***`[nombre_atributo]`***, selecciona los elementos que tienen establecido el atributo llamado `nombre_atributo`, independientemente de su valor.
- ***`[nombre_atributo=valor]`***, selecciona los elementos que tienen establecido un atributo llamado `nombre_atributo` con un valor igual a `valor`.
-

A continuación se muestran ***algunos ejemplos*** de estos tipos de selectores:

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

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

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

## METODOLOGÍA BEM EN CSS

En muchas ocasiones nos preguntamos qué nombre usar cuando vamos a crear una clase. Nuestro objetivo es sobre todo que sea intuitivo y que quede claro y ordenado.

Con el objetivo de escribir mejor código **CSS**, BEM conceptualiza las reglas CSS que escribimos en 3 elementos: **Bloque, Elemento, Modificador**.

- **Bloque**: Es un contenedor global que generalmente es una sección de nuestra web: **Header, Footer, Sidebar, Lista de articulos**, etc.
- **Elemento**: Es un elemento interno de un bloque por ejemplo: un **Menu** dentro de un **Header**. La forma que recomienda BEM para escribir un elemento es: **nombre-bloque__elemento{}.**
- **Modificador**: Representa un estado en particular de un bloque o elemento. Por ejemplo: **nombre-bloque–sidebar{}.**

Tratemos de mirar otro ejemplo para comprender un poco más como funciona la metodología BEM.

**HTML**

```
<div class="audi-a1">
 <div class="audi-
 a1__volante">
 </div>
 <div class="audi-
 a1__asientos">
 </div>
```

**CSS CON BEM**

```
/*
 *Audi A1
 */
 audi-a1{
 llantas:4;
 parabrisas:1;
 luces-delanteras: 2;
```

```
/*
 *Audi A1 Deportivo
 */
 audi-a1--deportivo{
 puertas: 4;
 quemacocos: 1;
 luces-neon: 4;
 }
```

```
</div>


<div class="audi-a1 audi-a1--deportivo">

<div class="audi-a1__volante">

</div>

<div class="audi-a1__asientos">

</div>

</div>
```

```
luces-traseras: 2;

puertas: 2;

}


audi-a1__volante{

material: plastico;

color: gris;

}


audi-a1__asientos{

material: tela;

color: gris;

}
```

```


audi-a1--deportivo audi-a1__volante{

material: piel;

color: negro;

}

audi-a1--deportivo audi-a1__asientos{

material: gamusa;

color: dorado;

}
```

## CAPÍTULO 3. MODELO DE CAJAS CON HTML5 Y CSS3

### Elementos block

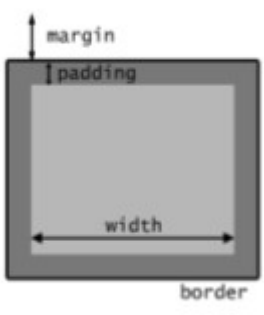
### (Se usará la práctica 1 como resultado final)

Con respecto a la estructura, básicamente cada navegador ordena los elementos por defecto de acuerdo a su tipo: *block* (bloque) o *inline* (en línea). Esta clasificación está asociada con la forma en que los elementos son mostrados en pantalla.

- **Elementos *block*** son posicionados uno sobre otro hacia abajo en la página.
- **Elementos *inline*** son posicionados lado a lado, uno al lado del otro en la mismalínea, sin ningún salto de línea a menos que ya no haya más espacio horizontalpara ubicarlos.

(VER PROPIEDAD DISPLAY EXPLICADA MÁS ADELANTE)

Casi todos los elementos estructurales en nuestros documentos serán tratados por los navegadores como elementos *block* por defecto. Esto significa que cada elemento HTML que representa una parte de la organización visual (por ejemplo, **<section>**, **<nav>**,**<header>**, **<footer>**, ** **) será posicionado debajo del anterior.



. Una página web es en realidad un grupo de cajas ordenadas siguiendo ciertas reglas. Estas reglas son establecidas por estilos provistos por los navegadores o por los diseñadores usando CSS.

**Selector universal ***

Comencemos con algunas reglas básicas que nos ayudarán a proveer consistencia al diseño:

```
* {
margin: 0px;
padding: 0px;
}
```

Normalmente, para la mayoría de los elementos, necesitamos personalizar los márgenes o simplemente mantenerlos al mínimo. Algunos elementos por defecto tienen márgenes que son diferentes de cero y en la mayoría de los casos demasiado amplios.

**Otra regla básica que debemos declarar desde el comienzo es la definición por defecto de** elementos estructurales de HTML5. Algunos navegadores aún no reconocen estos elementos o los tratan como elementos *inline* (en línea). Necesitamos declarar los nuevos elementos HTML5 como elementos *block* para asegurarnos de que serán tratados como regularmente se hace con elementos ** ** y de este modo construir nuestro modelo de caja:

```
header, section, footer, aside, nav, article, figure, figcaption, hgroup{
      display: block;
}
```

A partir de ahora, los elementos afectados por la regla **serán posicionados uno sobre otro a menos que especifiquemos algo diferente más adelante.**

**Por defecto, la etiqueta** `<body>` (como cualquier otro elemento *block*) tiene un valor de ancho establecido en 100%. Esto significa que el cuerpo ocupará el ancho completo de la ventana del navegador. Por lo tanto, para centrar la página en la pantalla necesitamos centrar el contenido dentro del cuerpo.

```
body {text-align: center;}
```

## Creando la caja principal

```
#agrupar {
width: 960px;        →La anchura de la caja será de 960 pixeles
margin: 15px auto;   →margen superior e inferior de 15 píxeles e izq y drcho automático
text-align: left;    →Alinea el texto a la izquierda(esto es necesario hacerlo porque
                     antes se le ha dicho en el body que sería centrado)
}
```

## La cabecera

Siguiendo la etiqueta de apertura del ** ** principal se encuentra el primer elemento estructural de HTML5:**<header>**.

Este elemento contiene el título principal de nuestra página web y estará ubicado en la parte superior de la pantalla.

```
#cabecera {
background: #FFFBB9;
border: 1px solid #999999;
padding: 20px;
}
```

## Barra de navegación

**<nav> es un elemento *block* por lo que será ubicado debajo del** elemento previo, su ancho por defecto será 100% por lo que será tan ancho como su padre (el ** ** principal), y (también por defecto) será tan alto como su contenido y los **márgenes predeterminados**

```
#menu {
background: #CCCCCC;
padding: 5px 15px;
}
```

Dentro de la barra de navegación hay una lista creada con las etiquetas **<ul>** y **<li>**. Por defecto, los ítems de una lista son posicionados unos sobre otros. Para cambiar este comportamiento y colocar cada opción del menú una al lado de la otra, usando el selector **#menu li**, y luego asignamos a todos ellos el estilo

**display: inline-block**

**Servirá** para convertirlos en lo que se llama **cajas *inline*.** A diferencia de los elementos *block*, los elementos afectados por el parámetro **inline-block** estandarizado en CSS3 no generan ningún salto de línea pero nos permiten tratarlos como elementos block y así declarar un valor de ancho determinado.

También eliminamos el pequeño gráfico generado por defecto por los navegadores delante de cada opción del listado con:

**list-style:none**

El estilo completo sería:

```
#menu li {
       display: inline-block;
       list-style: none;
       padding: 5px;
       font: bold 14px verdana, sans-serif;
}
```

**PROPIEDAD DISPLAY**

Esta propiedad nos permite establecer el tipo de caja que el navegador empleará para visualizar un elemento, siendo los tipos más comunes inline y block, aunque existen bastantes otros.

La sintaxis a emplear es del tipo:

selectorElemento {display: especificaciónDeVisualización; }

| PROPIEDAD CSS display | |
|---|---|
| **Función de la propiedad** | Permite definir el tipo de posición de caja para visualizar un elemento. |
| **Valor por defecto** | Depende del elemento (inline para elementos inline y block para elementos tipo block) |
| **Aplicable a** | Todos los elementos. |

| PROPIEDAD CSS display | |
|---|---|
| **Valores posibles para esta propiedad** | **inline** (el elemento se muestra en una caja inline) |
| | **block** (el elemento se muestra en una caja block). |
| | **none** (el elemento no se muestra; el efecto es como si no existiera, por lo que su espacio será ocupado por otros elementos) |
| | **list-item** (el elemento se comporta como si fuera un elemento li) |
| | **inline-block** (el elemento genera una caja block pero que se comporta como si fuera inline admitiendo otros elementos en la misma línea; el comportamiento se asemeja al de los elementos img) |
| | Otros que llevan a que el elemento simule el comportamiento de otro (inline-table, table, table-caption, table-cell, table-column, table-column-group, table-footer-group, table-header-group, table-row, table-row-group) |
| | Otros avanzados (flex, inline-flex, grid, inline-grid, run-in) |
| | **inherit** (se heredan las características del elemento padre). |
| **Ejemplos aprenderaprogramar.com** | #content1 {display: inline;}<br>.elementoMonter {display: block;} |

La propiedad display admite numerosos valores, pero los más usados son inline, block e inline-block.

## PROPIEDAD OVERFLOW

Normalmente, los contenidos de un elemento se pueden mostrar en el espacio reservado para ese elemento. Sin embargo, en algunas ocasiones el contenido de un elemento no cabe en el espacio reservado para ese elemento y se desborda.

La situación más habitual en la que el contenido sobresale de su espacio reservado es cuando se establece la anchura y/o altura de un elemento mediante la propiedad `width` y/o `height`. Otra situación habitual es la de las líneas muy largas contenidas dentro de un elemento `<pre>`, que hacen que la página entera sea demasiado ancha.

CSS define la propiedad `overflow` para controlar la forma en la que se visualizan los contenidos que sobresalen de sus elementos.

| | |
|---|---|
| Propiedad | **overflow** |
| Valores | visible \| hidden \| scroll \| auto \| inherit |
| Se aplica a | Elementos de bloque y celdas de tablas |

| | |
|---|---|
| Propiedad | **overflow** |
| Valor inicial | visible |
| Descripción | Permite controlar los contenidos sobrantes de un elemento |

Los valores de la propiedad `overflow` tienen el siguiente significado:

- **`visible`**: el contenido no se corta y se muestra sobresaliendo la zona reservada para visualizar el elemento. Este es el comportamiento por defecto.
- **`hidden`**: el contenido sobrante se oculta y sólo se visualiza la parte del contenido que cabe dentro de la zona reservada para el elemento.
- **`scroll`**: solamente se visualiza el contenido que cabe dentro de la zona reservada para el elemento, pero también se muestran barras de *scroll* que permiten visualizar el resto del contenido.
- **`auto`**: el comportamiento depende del navegador, aunque normalmente es el mismo que la propiedad `scroll`.

La siguiente imagen muestra un ejemplo de los tres valores típicos de la propiedad `overflow`:

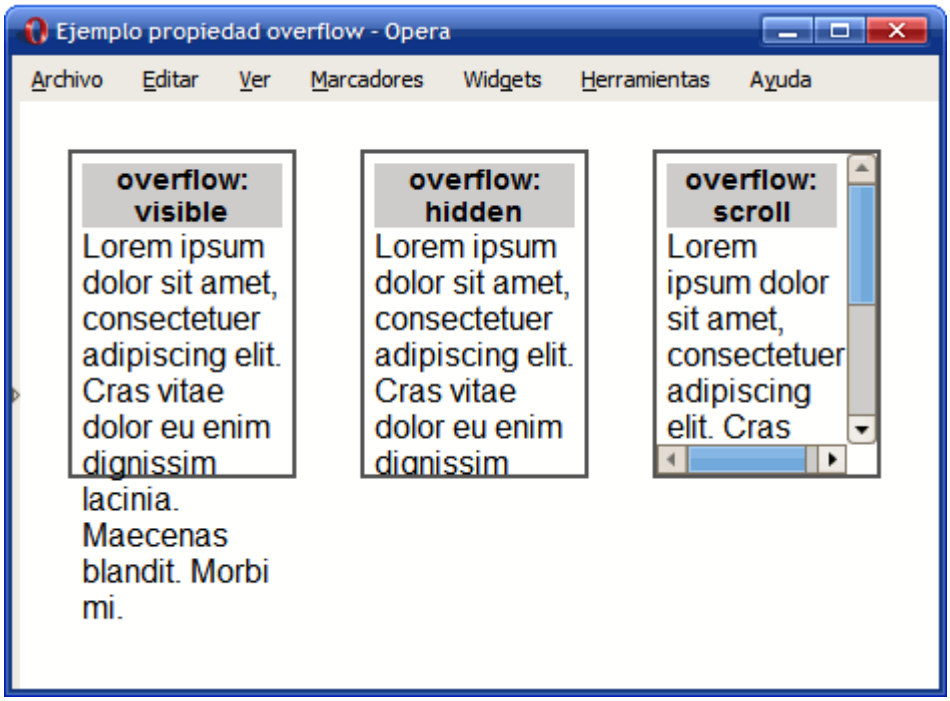



**Figura 5.23** Ejemplo de propiedad overflow

## Section y aside

**Los siguientes elementos estructurales en nuestro código son dos cajas ordenadas** horizontalmente. El Modelo de Caja Tradicional es construido sobre estilos CSS que nos permiten especificar la posición de

cada caja. Usando la propiedad **float** podemos **posicionar estas cajas del lado izquierdo o derecho de acuerdo a nuestras necesidades.**

```
#seccion {
	float: left;
	width: 660px;
	margin: 20px;
}
```

```
#columna {
	float: left;
	width: 220px;
	margin: 20px 0px;
	padding: 20px;
	background: #CCCCCC;
}
```

La propiedad de CSS **float h**ace que el elemento flote hacia un lado o al otro en el espacio disponible. Los elementos afectados por **float** actúan como elementos *block* (con la diferencia de que son ubicados de acuerdo al valor de esta propiedad y no el flujo normal del documento). Los elementos son movidos a izquierda o derecha en el área disponible, tanto como sea posible, respondiendo al valor de `float`.

### ***Footer***

```
#pie {
	clear: both;-->RESTAURA EL FLUJO DEL DOCUMENTO
	text-align: center;
	padding: 20px;
	border-top: 2px solid #999999;
}
```

**Clear** simplemente restaura las condiciones normales del área ocupada por el elemento, no permitiéndole posicionarse adyacente a una caja flotante. El valor usualmente utilizado es **both**, el cual significa que ambos lados del elemento serán restaurados y el elemento seguirá el flujo normal (este elemento ya no es flotante como los anteriores). Esto, para un elemento *block*, quiere decir que será posicionado debajo del **último elemento, en una nueva línea.**

## CAPITULO 4. NUEVAS PROPIEDADES CSS3

CSS3 provee ***nuevas propiedades*** para crear efectos visuales y dinámicos que son parte esencial de la web en estos días.

**border-radius** Esta propiedad genera esquinas redondeadas para la caja formada por el elemento. Posee dos parámetros diferentes que dan forma a la esquina. El primer parámetro determina la curvatura horizontal y el segundo la vertical, otorgando la posibilidad de crear una elipsis. Para declarar ambos parámetros de la curva, los valores deben ser separados por una barra (por ejemplo, **border-radius: 15px / 20px**). Usando solo un valor determinaremos la misma forma para todas las esquinas (por ejemplo, **border-radius: 20px**). Un valor para cada esquina puede ser declarado en un orden que sigue las agujas del reloj, comenzando por la esquina superior izquierda.

**box-shadow** Esta propiedad crea sombras para la caja formada por el elemento. Puede tomar cinco parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, el valor de difuminación, y la palabra clave **inset la cual** convierte a la sombra externa en una sombra interna, lo cual provee un

efecto de profundidad al elemento afectado. Los desplazamientos pueden ser negativos, y el valor de difuminación y el valor **inset** son opcionales (por ejemplo, **box-shadow: #000000 5px 5px 10px inset**).

**text-shadow** Esta propiedad es similar a **box-shadow** pero específica para textos. Toma cuatro parámetros: el color, el desplazamiento horizontal, el desplazamiento vertical, y el valor de difuminación (por ejemplo, **text-shadow: #000000 5px 5px 10px**).

**@font-face** Esta regla nos permite cargar y usar cualquier fuente que necesitemos. Primero, debemos declarar la fuente, proveer un nombre con la propiedad **font-family** y especificar el archivo con **src** (por ejemplo, **@fontface{ font-family: Mifuente; src: url('font.ttf') }**). Luego de esto, podremos asignar la fuente. ejemplo

```
#titulo {
font: bold 36px MiNuevaFuente, verdana, sans-serif;
text-shadow: rgb(0,0,150) 3px 3px 5px;
}

@font-face {
font-family: 'MiNuevaFuente';
src: url('font.ttf');
}
```

**linear-gradient(posición inicio, color inicial, color final)** Esta función puede ser aplicada a las propiedades **background** o **background-image** para generar un gradiente lineal. Los atributos indican el punto inicial y los colores usados para crear el gradiente. El primer valor puede ser especificado en pixeles, en porcentaje o usando las palabras clave **top**, **bottom**, **left** y **right**. El punto de inicio puede ser reemplazado por un ángulo para proveer una dirección específica para el gradiente .

```
background:linear-gradient(top, #FFFFFF, #006699);
```

**El siguiente ejemplo muestra una gradiante linear que empieza desde la izquierda. Empieza completamente transparente para llegar al color red.**

```
background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));
```

**Y para hacer una imagen transparente sería**

```
background: linear-gradient(rgba(255,255,255,0.2), rgba(255,255,255,0.2)),url (nombrefich.jpg) ;/
```

**En este caso, los valores 255 en el rgba van a conseguir poner transparente la imagen indicada, y el 4° valor del rgba servirá para indicar lo más o menos opaco que se quiera poner la imagen.**

**radial-gradient(posición inicio, forma, color inicial, color final)** Esta función puede ser aplicada a las propiedades **background** o **background-image** para generar un gradiente radial. La posición de inicio es

el origen y puede ser declarado en pixeles, porcentaje o como una combinación de las palabras clave **center**, **top**, **bottom**, **left** y **right**. Existen dos valores para la forma: **circle** y **ellipse**, y puntos de terminación pueden ser declarados para cada color indicando la posición donde la transición comienza (por ejemplo, **radial-gradient(center, circle, #FFFFFF 0%, #006699 200%);**).

**rgba()** Esta función es una mejora de **rgb()**. Toma cuatro valores: el color rojo (0-255), el color verde (0-255), el color azul (0-255), y la opacidad (un valor entre 0 y 1).

```
#titulo {
        font: bold 36px MiNuevaFuente, verdana, sans-serif;
        text-shadow: rgba(0,0,0,0.5) 3px 3px 5px;
}
```

**hsla()** Esta función es una mejora de **hsl()**. Puede tomar cuatro valores: el tono (un valor entre 0 y 360), la saturación (un porcentaje), la luminosidad (un porcentaje), y la opacidad (un valor entre 0 y 1).

**outline :** generan una especie de segundo borde , lo que se llama un perfil,alejado del borde original del elemento (por ejemplo, **outline: 1px solid #000000;** ).

**Outline-style:**establece el estilo del perfil de los elementos. (por ejemplo:outline-style: dotted ) otros valores serían:none,dashed, solid,double, groove

**border-image** Esta propiedad crea un borde con una imagen personalizada. Necesita que el borde sea declarado previamente con las propiedades **border** o **border-with**, y toma al menos **tres parámetros**: la URL de la imagen, el tamaño de las piezas que serán tomadas de la imagen para construir el borde, y una palabra clave que especifica cómo esas piezas serán ubicadas alrededor del elemento (por ejemplo, **border-image: url("file.png") 15 stretch;**).

**Nota:se mostrará la imagen dependiendo del grosor que se le ponga al border, no olvidar esto.**

**transform** Esta propiedad modifica la forma de un elemento**. Utiliza cuatro funciones básicas**: **scale** (escalar), **rotate** (rotar), **skew** (inclinar), y **translate** (trasladar o mover). La función **scale** recibe solo un parámetro. Un valor negativo invierte el elemento, valores entre 0 y 1 reducen el elemento y valores mayores que 1 expanden el elemento (por ejemplo, **transform: scale(1.5);**). La función **rotate** usa solo un parámetro expresado en grados para rotar el elemento (por ejemplo, **transform: rotate(20deg);**). La función **skew** recibe dos valores, también en grados, para la transformación horizontal y vertical (por ejemplo, **transform: skew(20deg, 20deg);**). La función **translate** mueve el objeto tantos pixeles como sean especificados por sus parámetros (por ejemplo, **transform: translate(20px);**).

**transition** Esta propiedad puede ser aplicada para crear una transición entre dos estados de un elemento. Recibe hasta cuatro parámetros: la propiedad afectada, el tiempo que le tomará a la transición desde el comienzo hasta el final, una palabra clave para especificar cómo la transición será realizada (**ease**, **linear**, **ease-in**, **ease-out**, **ease-in-out**) y un valor de retardo que determina el tiempo que la transición tardará en comenzar (por ejemplo, **transition: color 2s linear 1s;**)

**background**

Un ejemplo de la propiedad background con sus atributos podría ser:

```
body {
```

```
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
    background-position: right top;-→para indicar justo el lugar donde se situa la imagen indicada
    margin-right: 200px;-→margen derecho de todo el body que ayudaría a parar el texto justo para no pisar la imagen.
    background-attachment: fixed; para que la imagen no se mueva aunque se usara la barra de desplazamiento
}
```

# PREFIJOS CSS DE NAVEGADORES

Se llaman prefijos de navegador o prefijos comerciales (vendor prefixes) a un prefijo que se antepone a una regla CSS destinado a que dicha regla sea leída y aplicada exclusivamente por un navegador concreto (por ejemplo Chrome) pero no por el resto de navegadores. El uso de prefijos suele aplicarse a propiedades que se encuentran en fase experimental o que aún no se han convertido en un estándar.

| Prefijo | Familia de navegadores a los que aplica | |
|---|---|---|
| **-webkit-** | Chrome, Safari, Android, iOs | |
| **-moz-** | Firefox | |
| **-o-** | Opera | |
| **-ms-** | Microsoft Internet Explorer | |

**Ejemplo.**

```
#elemento{
  background: -webkit-linear-gradient(red, blue); /* For Safari */
  background: -o-linear-gradient(red, blue); /* For Opera*/
  background: -moz-linear-gradient(red, blue); /* For Firefox*/
  background: -ms-linear-gradient(red, blue); /* For Internet Explorer*/
  background: linear-gradient(red, blue); /* Standard syntax */
}
```

Para evitar tener que hacer esto, y ser más productivo nuestro código, se puede utilizar **–Prefix-Free.** Es un **fichero en javascript** que automáticamente nos agrega estos prefijos a las propiedades css que lo necesiten, simplificando el código ya que sólo debemos escribir la propiedad una vez, por tanto nos ahorra mucho tiempo y hace nuestro código css más fácil de entender.

Es compatible con IE9+, Opera 10+, Firefox 3.5+, Safari 4+ y Chrome en escritorio. En móviles con Mobile Safari, Android browser, Chrome and Opera Mobile.

# Capítulo 5. Pseudo-clases

**La pseudo-clase :first-child**

Selecciona el primer elemento hijo de un elemento.

Ejemplo:

```
p:first-child {  color: red;}
```

tendrá el color rojo el primer párrafo de cada elemento

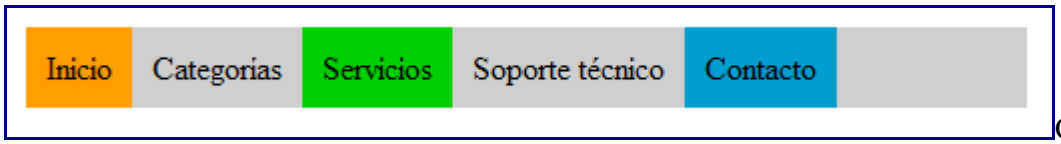
**Entonces las pseudoclases first-child, last-child y nth-child se comportarán de la siguiente manera:**

Ejemplo:

Vamos a crear el siguiente menú:

```
<ul id="menu">
    <li><a href="#">Inicio</a></li>
    <li><a href="#">Categorías</a></li>
    <li><a href="#">Servicios</a></li>
    <li><a href="#">Soporte técnico</a></li>
    <li><a href="#">Contacto</a></li>
</ul>
```

Y le vamos a dar el siguiente estilo sencillo utilizando los selectores que estamos comentando:




Como puedes observar, le estamos dando un estilo diferente (background) al primer elemento del menú, al último y a otro seleccionado. Esto lo conseguimos añadiendo lo siguiente en nuestra hoja de estilos:

```
#menu li:first-child {
    background-color:#FF9900;
}

#menu li:last-child {
    background-color: #0099CC;
}

#menu li:nth-child(3) {
    background-color: #00CC00;
}
```

**Las pseudo-clases :link y :visited**

Las pseudo-clases :link y :visited se pueden utilizar para aplicar diferentes estilos a los enlaces de una misma página:

- :link se aplica a todos los enlaces que todavía no han sido visitados por el usuario.
- :visited se aplica a todos los enlaces que han sido visitados al menos una vez por el usuario.

```
a:link    { color: red;   }
a:visited { color: green; }
```

**Las pseudo-clases :hover, :active y :focus**

Las pseudo-clases :hover, :active y :focus se pueden aplicar a cualquier elemento y no sólo a los enlaces como en el caso de :link y :visited

- :hover, se activa cuando el usuario pasa el ratón o cualquier otro elemento apuntador por encima de un elemento.
- :active, se activa cuando el usuario activa un elemento, por ejemplo cuando pulsa con el ratón sobre un elemento. El estilo se aplica durante un espacio de tiempo prácticamente imperceptible, ya que sólo dura desde que el usuario pulsa el botón del ratón hasta que lo suelta.
- :focus, se activa cuando el elemento tiene el foco del navegador, es decir, cuando el elemento está seleccionado. Normalmente se aplica a los elementos <input> de los formularios cuando están activados y por tanto, se puede escribir directamente en esos campos.

De las definiciones anteriores se desprende que un mismo elemento puede verse afectado por varias pseudo-clases diferentes de forma simultánea. Cuando se pulsa por ejemplo un enlace que fue visitado previamente, al enlace le afectan las pseudo-clases :visited, :hover y :active.

**El pseudo-elemento :first-line**

El pseudo-elemento :first-line permite seleccionar la primera línea de texto de un elemento. Así, la siguiente regla CSS muestra en mayúsculas la primera línea de cada párrafo:

```
p:first-line { text-transform: uppercase; }
```

Este pseudo-elemento sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

Se pueden combinar varios pseudo-elementos de tipo :first-line para crear efectos vanzados:

```
div:first-line { color: red; }
p:first-line { text-transform: uppercase; }

<div>
```

```
 <p>Lorem ipsum dolor sit amet...</p>
 <p>Lorem ipsum dolor sit amet...</p>
 <p>Lorem ipsum dolor sit amet...</p>
</div>
```

En el ejemplo anterior, la primera línea del primer párrafo también es la primera línea del elemento , por lo que se le aplican las dos reglas CSS y su texto se ve en mayúsculas y de color rojo.

**El pseudo-elemento :first-letter**

El pseudo-elemento :first-letter permite seleccionar la primera letra de la primera línea de texto de un elemento. De esta forma, la siguiente regla CSS muestra en mayúsculas la primera letra del texto de cada párrafo:

```
p:first-letter { text-transform: uppercase; }
```

Los signos de puntuación y los caracteres como las comillas que se encuentran antes y después de la primera letra también se ven afectados por este pseudo-elemento.

Este pseudo-elemento sólo se puede utilizar con los elementos de bloque y las celdas de datos de las tablas.

**Los pseudo-elementos :before y :after**

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: "."; }
```

**Pseudoclase CSS cursor.**

Aunque CSS nos ofrezca la posibilidad de cambiar el aspecto del cursor del ratón, esto es una acción que tenemos que llevar a cabo con bastante cuidado, desde el punto de vista de la usabilidad. Debemos saber que las personas están acostumbradas a visualizar ciertos tipos de cursor en ciertas ocasiones.

una la lista de los cursores disponibles por defecto en Windows, y que también tendremos disponibles con CSS.

- default (flecha)
- crosshair (cruz)
- e-resize (flecha que apunta a la derecha)
- hand (mano)
- help (signo de pregunta)
- move (cruz con flechas en los extremos)
- n-resize (flecha que apunta hacia arriba)
- ne-resize (flecha que apunta al noreste)
- nw-resize (flecha que apunta al noroeste)
- pointer (mano)

- s-resize (flecha que apunta hacia abajo)
- se-resize (flecha que apunta hacia el sudeste)
- sw-resize (flecha que apunta hacia el sudoeste)
- text (I-beam)
- w-resize (flecha que apunta a la izquierda)
- wait (reloj de arena)

Ejemplos:

- body {cursor: pointer }-→en todo el cuerpo de la página el cursor del ratón será una manita.
- .boton:active{

  **background-color**: #000;
  cursor: wait

  }-→cuando el botón es pulsado , el cursor se convierte en un reloj de arena

## Capítulo 6.Formularios Web

**Los formularios constituyen el principal medio de comunicación entre usuarios y aplicaciones web. HTML5 incorpora nuevos** tipos para el elemento **<input>**, una API completa para validar y procesar formularios, y atributos para mejorar esta interface.

# Tipos

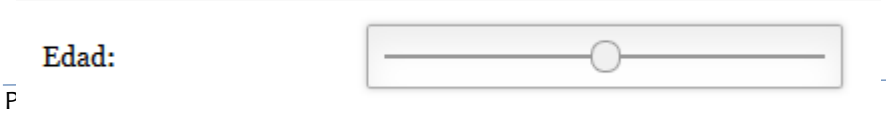
Algunos de los nuevos tipos de campo introducidos por HTML5 tienen condiciones de validación implícitas. Otros solo declaran un propósito para el campo que ayudará a los navegadores a presentar el formulario en pantalla.
**email :**Este tipo de campo valida la entrada como una dirección de email.
**search** Este tipo de campo da información al navegador sobre el propósito del campo (búsqueda) para ayudar a presentar el formulario en pantalla.

**url** Este tipo de campo valida la entrada como una dirección web.

**tel** Este tipo de campo da información al navegador sobre el propósito del campo (número telefónico) para ayudar a presentar el formulario en pantalla.

**number** Este tipo de campo valida la entrada como un número. Puede ser combinado con otros atributos (como **min, max y step) para limitar los números permitidos.**


**range** Este tipo de campo genera un nuevo control en pantalla para la selección de números. La entrada es limitada por los atributos **min, max** y **step**. El atributo **value** establece el valor inicial para el elemento.



**date** Este tipo de campo valida la entrada como una fecha en el formato **año-mes-día**.

**month** Este tipo de campo valida la entrada como una fecha en el formato **año-mes**.

**week** Este tipo de campo valida la entrada como una fecha en el formato **año-semana** donde el segundo valor es representado por una letra W y el número de la semana.

**time** Este tipo de campo valida la entrada como una hora en el formato **hora:minutos:segundos**. También puede tomar otras sintaxis como **hora:minutos**.

**datetime** Este tipo de campo valida la entrada como fecha y hora completa, incluyendo zona horaria.

**datetime-local** Este tipo de campo valida la entrada como una fecha y hora completa, sin zona horaria.

**color** Este tipo de campo valida la entrada como un valor de color.

## Atributos

Nuevos atributos fueron también agregados en HTML5 para mejorar la capacidad de los formularios y ayudar al control de validación.
**autocomplete** Este atributo especifica si los valores insertados serán almacenados para futura referencia. Puede tomar dos valores: **on** y **off**.
**autofocus** Este es un atributo booleano que enfoca el elemento en el que se encuentra cuando la página es cargada.
**novalidate** Este atributo es exclusivo para elementos **<form>**. Es un atributo booleano que establece si el formulario será validado por el navegador o no.
**formnovalidate** Este atributo es exclusivo para elementos de formulario individuales. Es un atributo booleano que establece si el elemento será validado por el navegador o no.
**placeholder** Este atributo ofrece información que orientará al usuario sobre la entrada esperada. Su valor puede ser una palabra simple o un texto corto, y será mostrado como una marca de agua dentro del campo hasta que el elemento es enfocado.

**required** Este atributo declara al elemento como requerido para validación. Es un atributo booleano que no dejará que el formulario sea enviado hasta que una entrada para el campo sea provista.

**pattern** Este atributo especifica una expresión regular contra la cual la entrada será validada.

**multiple** Este es un atributo booleano que permite ingresar múltiples valores en el mismo campo (como múltiples cuentas de email, por ejemplo). Los valores deben ser separados por coma.

**form** Este atributo asocia el elemento al formulario. El valor provisto debe ser el valor del atributo **id** del elemento **<form>**.

**list** Este atributo asocia el elemento con un elemento **<datalist>** para mostrar una lista de posibles valores para el campo. El valor provisto debe ser el valor del atributo **id** del elemento **<datalist>**.

## Elementos

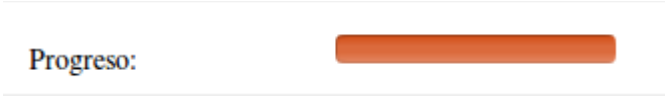
HTML5 también incluye nuevos elementos que ayudan a mejorar y expandir formularios.
**<datalist>** Este elemento hace posible incluir una lista de opciones predefinidas que será mostrada en un elemento **<input>** como valores sugeridos. La lista es construida con el elemento **<option>** y cada opción es declarada con los atributos **value** y **label**. Esta lista de opciones se relaciona con un elemento **<input>** por medio del atributo **list**.

```
<datalist id="informacion">
        <option value=”123123123” label=”Teléfono 1”>
        <option value=”456456456” label=”Teléfono 2”>
</datalist>

<input type="tel" name="telefono" id="telefono" list="informacion">
```

**<progress>** Este elemento representa el estado en la evolución de una tarea (por ejemplo, una descarga).

```
<progress value="10" >
```




**<meter>** Este elemento representa una medida, como el tráfico de un sitio web.

**<output>** Este elemento presenta un valor de salida para aplicaciones dinámicas.

# CAPITULO 7. VÍDEO Y AUDIO

**Elementos**
HTML5 provee dos nuevos elementos HTML para procesar medios y una API específica para acceder a la librería de medios.
**<video>** Este elemento nos permite insertar un archivo de video en un documento HTML.
**<audio>** Este elemento nos permite insertar un archivo de audio en un documento HTML.
**Atributos**
La especificación también provee atributos para los elementos **<video>** y **<audio>**:

**src** Este atributo declara la URL del medio a ser incluido en el documento. Puede usar el elemento **<source>** para proveer más de una fuente y dejar que el navegador elija cual reproducir.
**controls** Este atributo, si está presente, activa los controles por defecto. Cada navegador provee sus propias funciones, como botones para reproducir y pausar el medio, así como barra de progreso, entre otras.

**autoplay** Este atributo, si está presente, le indicará al navegador que comience a reproducir el medio lo más pronto posible.
**loop** Este atributo hará que el navegador reproduzca el medio indefinidamente.
**preload** Este atributo recomienda al navegador qué hacer con el medio. Puede recibir tres valores diferentes: **none**, **metadata** y **auto**. El valor **none** le dice al navegador que no descargue el archivo hasta que el usuario lo ordene. El valor **metadata** le recomienda al navegador descargar información básica sobre el medio. El valor **auto** le dice al navegador que comience a descargar el archivo tan pronto como sea posible.

### Atributos de video

Existen algunos atributos que son específicos para el elemento **`<video>`**:
**poster** Este atributo provee una imagen para mostrarla en lugar del video antes de ser reproducido.
**width** Este atributo determina el tamaño del video en pixeles.
**height** Este atributo determina el tamaño del video en pixeles.

---

```
<video id="medio" width="720" height="400" preload controls loop
poster="http://lolo.com/content/poster.jpg">
        <source src="http://www.lolo.com/content/trailer.mp4">
        <source src="http://www.lolo.com/content/trailer.ogg">
</video>

</section>
```

---

### Métodos

HTML5 incluye una API específica para formularios que provee métodos, eventos y propiedades. Algunos de los métodos son:
**setCustomValidity(mensaje)** Este método nos permite declarar un error y proveer un mensaje de error para un proceso
de validación personalizado. Para anular el error, debemos llamar al método con una cadena de texto vacía como
atributo.
**checkValidity()** Este método solicita al navegador iniciar el proceso de validación. Activa el proceso de validación
provisto por el navegador sin la necesidad de enviar el formulario. Este método retorna **true** (verdadero) si el
elemento es válido.

### Eventos

Los eventos incorporados para esta API son los siguientes:
**invalid** Este evento es disparado cuando un elemento inválido es detectado durante el proceso de validación.
**forminput** Este evento es disparado cuando un formulario recibe la entrada del usuario.
**formchange** Este evento es disparado cuando un cambio ocurre en el formulario.

**CSS AVANZADO**
Imagina que deseas poner el típico enlace "leer más" en el que al pulsar en el se expande un párrafo para poder leer más sobre ese tema.
Fíjate en este código

Usa este sencillo **código html** teniendo en cuenta que el **for del label** debe ser el **id del checkbox**

```
<input type="checkbox" class="leermas" id="micheck" />

<p class="recoger">Aquí aparecera el texto de mi segundo mensajeLorem ipsum dolor sit amet, consectetur adipisicing elit.<span class="mastexto">
Aspernatur minus, eum mollitia doloribus voluptatum ducimus est harum dolore veritatis quaerat ea hic. Dolorum, natus. Placeat ipsa maiores imquod ullam provident!
</p>

<label for="micheck" class="estirar"></label>
```

Luego, **con el código css,** debemos hacer que aparezca el texto del label correspondiente en cada momento (hasta ahora está vacío), es decir, "leer mas" o "leer menos".
Y a la vez, tendrás que hacer que aparezca o no el texto correspondiente.

**Fíjate en este código**

```
.leermas {
  display: none;
}
.leermas:checked ~ .recoger .mastexto {
  opacity: 1;
  font-size: inherit;
  max-height: 999em;
}

.leermas ~ .estirar:before {
  content: 'Show more';
}

.leermas:checked ~ .estirar:before {
  content: 'Show less';
}

.estirar {
  cursor: pointer;
  display: inline-block;
  padding: 0 .5em;
  color: #666;
  font-size: .9em;
  line-height: 2;
  border: 1px solid #ddd;
  border-radius: .25em;
}
```