

Lenguaje **DHTML**

Creación y uso de Hojas de Estilo

- Definición de Hojas de Estilo con la etiqueta <STYLE>
- Definición de Hojas de Estilo en Ficheros Externos
- Definición de Clases de Estilo
- Definición de Estilos Individuales con Nombre
- Uso de Criterios de Selección Contextual
- Especificación de Estilos para Elementos Individuales
- Combinando Hojas de Estilo

Propiedades de Formato de Elementos de Bloque

- Formato de Bloques: Introducción y Ejemplos
- Márgenes
- Bordes
- Separadores
- Herencia de Propiedades de Formato de Bloque

Manual de Referencia de Hojas de Estilo

- Comentarios en las Hojas de Estilo
- Nuevas etiquetas HTML
- Nuevos atributos para las etiquetas HTML

Propiedades de las Hojas de Estilo

Unidades

Creación y hojas de Estilo

Definición de Hojas de Estilo con la etiqueta <STYLE>

Para definir una hoja de estilo directamente dentro de un documento se utiliza la etiqueta `<style>` dentro de la sección `<head> ... </head>`. La etiqueta `<style>` abre la hoja de estilo, y la etiqueta `</style>` la cierra. Asegúrate de utilizar `<style>` antes de `<body>`. Cuando use `<style>` podrá especificar el atributo `type` para indicar que tipo de sintaxis se va a emplear. Los dos posibles son `"text/css"` y `"text/javascript"`. EL valor por defecto es `"text/css"`. El siguiente ejemplo define una hoja de estilo que especifica que todos los títulos de nivel 4 serán en mayúsculas y azules, y todos los bloques en cursiva y rojos:

Sintaxis CSS

```
<head>
  <style type="text/css">
    h4 {text-transform: uppercase; color: blue;}
    blockquote {font-style: italic; color: red;}
  </style>
</head>
<body>
  ...
</body>
```

Sintaxis JavaScript

```
<head>
  <style type="text/javascript">
    tags.h4.textTransform = "uppercase"
    tags.h4.color = "blue"
    tags.blockquote.fontStyle = "italic"
    tags.blockquote.color = "red"
  </style>
</head>
<body>
```

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

Uso de la hoja de estilo

```
<h4>Este titulo es azul y esta en mayúsculas.</h4>
<blockquote>Este bloque en cursiva es rojo.</blockquote>
```

Resultado del ejemplo

ESTE TITULO ES AZUL Y ESTA EN MAYÚSCULAS.

Este bloque en cursiva es rojo.

Definición de Hojas de Estilo en Ficheros Externos

Se puede definir una hoja de estilo en un fichero distinto del que contiene la página y después enlazarlos. Las ventajas de este método son que podremos utilizar la hoja de estilo desde cualquier documento HTML. Se podría pensar en una hoja de estilo así definida como en un patrón que pudiera aplicarse a cualquier documento. De esta forma, se puede aplicar un estilo a todas las páginas servidas desde un sitio Web sin más que incluir un enlace al fichero con el estilo en cada página.

La sintaxis para definir estilos en ficheros es idéntica a la que se usa para definirlos en el propio documento, excepto que no es necesario incluir la etiqueta `<style>`. He aquí un ejemplo:

Sintaxis CSS

```
/* hoja de estilo externa estilo.html */
all.BOLDBLUE {color: blue; font-weight: bold;}
h1 {line-height: 18pt;}
p {color: yellow;}
/* fin de fichero */
```

Sintaxis JavaScript

```
/* hoja de estilo externa estilo.html */
tags.BOLDBLUE.all.color = "blue";
tags.BOLDBLUE.all.fontWeight = "bold";
tags.h1.lineHeight = "18pt";
tags.p.color = "yellow";
/* fin de fichero */
```

Para utilizar esta hoja de estilo, se usa la etiqueta `<link>` como se muestra en el siguiente ejemplo:

Sintaxis CSS

```
<head>
  <title>El titulo</title>
  <link rel = stylesheet type = "text/css" href = "estilo.html">
```

```
</head>
```

Sintaxis JavaScript

```
<head>
  <title>El titulo</title>
  <link rel = stylesheet type = "text/javascript" href = "estilo.html">
</head>
```

Definición de Clases de Estilo

Si un documento incluye o se enlaza con una hoja de estilo, todos los estilos definidos en dicha hoja podrán utilizarse en cualquier punto del documento. Si la hoja de estilo especifica el estilo de una etiqueta HTML, entonces todas las etiquetas de ese tipo en el documento utilizarán dicho estilo. Puede haber casos en que interese aplicar un estilo selectivamente. Por ejemplo, se puede querer que los párrafos de un documento sean unas veces rojos y otras veces azules. En este caso definir un estilo que se aplique a todos los párrafos no será la solución correcta. Podemos obtener el efecto deseado definiendo una clase de estilo y especificando cuando queremos que sea utilizada. Para aplicar una clase de estilo a un elemento HTML, primero se debe definir la clase en la hoja de estilo, y después se utilizará empleando el atributo `class` en cualquier elemento.

Sintaxis CSS

```
<style type="text/css">
  all.GREENBOLD {color: green; font-weight: bold;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  classes.GREENBOLD.all.color = "green";
  classes.GREENBOLD.all.fontWeigth = "bold";
</style>
```

Uso de la hoja de estilo

```
<h3 class=GREENBOLD>Este titulo es muy verde</h3>
<p class=GREENBOLD>
  Este párrafo usa la clase de estilo GREENBOLD. Se puede utilizar el atributo class para especificar una clase de estilo para
  cualquier elemento HTML.
</p>
<blockquote class=GREENBOLD>
  Este bloque usa la clase de estilo GREENBOLD. En consecuencia, es verde y está en negrita. Puede ser útil para hacer que los
  bloques resalten del resto de la página.
</blockquote>
```

Resultado del ejemplo

Este titulo es muy verde

Este párrafo usa la clase de estilo GREENBOLD. Se puede utilizar el atributo class para especificar una clase de estilo para cualquier elemento HTML.

Este bloque usa la clase de estilo GREENBOLD. En consecuencia, es verde y está en negrita. Puede ser útil para hacer que los bloques resalten del resto de la página.

Con sintaxis JavaScript no se pueden utilizar guiones "-". La razón es que el guión es un operador de JavaScript. Los nombres de clases no pueden contener tampoco operadores como: -, +, *, /, %, . Cuando se definen clases de estilo se puede especificar a qué elementos se podrán aplicar dicha clase, o usaremos la palabra clave `all` para indicar que todos los elementos podrán utilizarla. En el siguiente ejemplo se crea una clase de estilo amarillo que podrá utilizar cualquier elemento HTML. También se crea una clase rojo que sólo podrán utilizar párrafos y bloques.

Sintaxis CSS

```
<style type="text/css">
  all.amarillo {color: yellow; font-weight: bold;}
  p.rojo {color: red; font-weight: bold;}
  blockquote.rojo {color: red; font-weight: bold;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  tags.amarillo.all.color = "yellow";
  tags.amarillo.all.fontWeight = "bold";
  tags.rojo.p.color = "red";
  tags.rojo.p.fontWeight = "bold";
  tags.rojo.blockquote.color = "red";
  tags.rojo.blockquote.fontWeight = "bold";
</style>
```

Uso de la hoja de estilo

```
<p class="rojo">Este párrafo es rojo.</p>
<p>Este párrafo es del color por defecto, porque no utiliza la clase rojo</p>
<blockquote class="rojo">Este bloque usa la clase rojo.</blockquote>
<h4 class="rojo">
  Este título intenta usar la clase rojo, pero no le está permitido
</h4>
<p class="amarillo">Este párrafo es amarillo</p>
<h4 class="amarillo">
  Este título es amarillo porque usa la clase amarillo
</h4>
```

Resultado del ejemplo

Este párrafo es rojo.

Este párrafo es del color por defecto, porque no utiliza la clase rojo

Este bloque usa la clase rojo.

Este título intenta usar la clase rojo, pero no le está permitido

Este párrafo es amarillo

Este título es amarillo porque usa la clase amarillo

Un elemento HTML sólo puede utilizar una clase de estilo. Si se especifican dos o más clases, se utilizará la primera. Por ejemplo, en el siguiente código un párrafo intenta usar las clases rojo y amarillo. Como resultado final se acaba empleando la clase rojo que es la primera que se especifica.

Ejemplo:

```
<p class="rojo" class="amarillo">Otro párrafo rojo.</p>
```

Resultado:

Otro párrafo rojo.

Definición de Estilos Individuales con Nombre

Se pueden crear estilos individuales con nombre. Los elementos HTML pueden utilizar un clase de estilo y un estilo individual con nombre. Normalmente estos se utilizan para expresar excepciones de estilo. Por ejemplo, si un párrafo utiliza la clase de estilo PRINCIPAL, podemos usar el estilo con nombre AZUL1 para expresar alguna diferencia respecto a la clase PRINCIPAL.

También son útiles para definir capas de contenidos HTML posicionadas de forma precisa. Parar definir estilos con nombre, en sintaxis CSS, se precede el nombre con el signo #. En JavaScript se utiliza la propiedad ids. Para aplicar el estilo a un elemento, se utiliza el nombre de estilo individual como valor del atributo ID. En el siguiente código se define una clase de estilo PRINCIPAL. Esta clase especifica una fuente de 15 puntos, negrita y de color rojo, y una interlinea de 20 puntos. También se define un estilo con nombre llamado AZUL1 cuyo color es azul.

Sintaxis CSS

```
<style type="text/css">
all.PRINCIPAL
{
    line-height: 20pt;
    font-size: 15pt;
    font-weight: bold;
    color: red;
}
```

```
}  
#AZUL1 {color: blue;}  
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">  
  with(classes.PRINCIPAL.all)  
  {  
    lineHeight = "20pt";  
    fontSize = "15pt";  
    fontWeight = "bold";  
    color = "red";  
  }  
  ids.AZUL1.color = "blue";  
</style>
```

Uso de la hoja de estilo

```
<p class="PRINCIPAL">  
  Aquí se puede ver un texto rojo y en negrita. En este párrafo la interlinea y el tamaño de la fuente son mayores de lo habitual.  
</p>  
<p class="PRINCIPAL" id="AZUL1">  
  Este párrafo es casi igual al anterior. Está en negrita y su fuente es mayor de lo habitual. Aunque usa la clase  
  PRINCIPAL es azul porque se utiliza el estilo con nombre AZUL1.  
</p>
```

Resultado del ejemplo

Aquí se puede ver un texto rojo y en negrita. En este párrafo la interlinea y el tamaño de la fuente son mayores de lo habitual.

Este párrafo es casi igual al anterior. Esta en negrita y su fuente es mayor de lo habitual. Aunque usa la clase PRINCIPAL es azul porque se utiliza el estilo con nombre AZUL1.

Uso de Criterios de Selección Contextual

Se pueden definir estilos para utilizarse con todos los elementos HTML de una clase particular. Si se necesita un mayor control sobre el uso de estilos podemos hacer que estos se apliquen selectivamente. Podríamos, por ejemplo, querer que el texto enfatizado sea de color verde, pero sólo si el texto

enfaticado está en el interior de un título de tamaño 4. Se puede conseguir este nivel de control sobre la aplicación de estilos usando los criterios de selección contextual. Estos, en general, permiten especificar que un estilo se aplicará sólo si un elemento se encuentra anidado dentro de un elemento de otro cierto tipo. Mediante la sintaxis, CSS esto se consigue listando ordenadamente los elementos HTML antes de las llaves. Con sintaxis JavaScript es necesario utilizar el método `contextual()`.

Sintaxis CSS

```
<style type="text/css">
  h4 em {color: green;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  contextual(tags.h4, tags.em).color = "green";
</style>
```

Uso de la hoja de estilo

```
<h4>El <em>texto enfaticado</em> de este titulo es verde.</h4>
<p>En cambio este <em>texto enfaticado</em> no es verde.</p>
```

Resultado del ejemplo

El **texto enfaticado** de este titulo es verde.

En cambio *este texto enfaticado* no es verde.

Ahora se verá otro ejemplo que hace que las marcas de los elementos de la lista que hereden de al menos dos listas desordenadas sean de color azul.

Sintaxis CSS

```
ul ul li {color: blue;}
```

Sintaxis JavaScript

```
contextual(tags.ul, tags.ul, tags.li).color = "blue";
```

Se pueden utilizar los criterios de selección contextual para buscar etiquetas, clases, ids o combinaciones de estos. En el siguiente ejemplo se crea la clase MAGENTA que lo colorea todo de magenta. Todos los párrafos MAGENTA que estén dentro de un `<div>` estarán en cursiva. Además los textos dentro de `` anidados dentro de párrafos dentro de un `<div>` en MAGENTA usarán una fuente grande.

Sintaxis CSS

```
<style type="text/css">
  all.MAGENTA {color: magenta;}
  div p.MAGENTA {font-style: italic;}
  div p.MAGENTA b {font-size: large;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  classes.MAGENTA.all.color = "magenta";
  contextual(tags.div, classes.MAGENTA.p).fontStyle = "italic";
  contextual(tags.div, classes.MAGENTA.p, tags.b).fontSize = "large";
</style>
```

Uso de la hoja de estilo

```
<div class=MAGENTA>
<h4> Titulo h4 en MAGENTA</h4>
<p>Este párrafo debería ser magenta y cursivo. Ahora viene un <b>texto grande</b>. Conseguimos este efecto con selección
  contextual</p>
</div>
<p class="MAGENTA">Este párrafo todavía es MAGENTA, pero como no está dentro de un bloque <div>, no es cursivo.</p>
```

Resultado del ejemplo

Titulo h4 en MAGENTA

*Este párrafo debería ser magenta y cursivo. Ahora viene un **texto grande**. Conseguimos este efecto con selección contextual*
Este párrafo todavía es MAGENTA, pero como no esta dentro de un bloque <div>, no es cursivo.

Especificación de Estilos para Elementos Individuales

De igual forma que se pueden definir hojas de estilo, podemos utilizar el atributo `style` de cualquier etiqueta HTML para definir un estilo que le será aplicado solamente a ella. Esta aproximación puede ser útil en situaciones en que necesitemos utilizar un estilo en un sitio y no sea necesario volver a utilizarlo. Sin embargo, en general, es mejor definir todos los estilos usados en un documento en un único lugar. Así es más fácil realizar modificaciones en su estilo sin tener que recorrerlo. Si se necesita hacer algún cambio sólo es necesario hacerlos una vez y el cambio automáticamente se aplica a todo el documento. A veces, sin embargo, se necesita especificar el estilo de un elemento y la forma más fácil de hacerlo es mediante el atributo `style`. En el ejemplo se especifica un estilo para el elemento `<p>`. También se muestra el uso de `` para aplicar un estilo a varios elementos.

Sintaxis CSS

```
<p style="color: green; font-weight: bold; margin-right: 20%;
```

```

        margin-left: 20%; border-width: 2pt; border-color: blue;">
    Este párrafo, y sólo este párrafo, es verde, esta en negrita y tiene un borde azul.
</p>
<p>
    Este párrafo es del color habitual, pero esta <span
        style="color: red; font-style: italic;">palabra</span> es diferente al resto.
</p>

```

Sintaxis JavaScript

```

<p style="color = 'green'; fontWeight = 'bold';
    marginRight = '20%'; marginLeft = '20%';
    borderWidth = '2pt'; borderColor = 'blue';">
    Este párrafo, y sólo este párrafo, es verde, esta en negrita, tiene unos grandes márgenes y un borde azul.
</p>
<p>
    Este párrafo es del color habitual, pero esta <span
        style="color = "red"; fontStyle = "italic";">palabra</span>
        es diferente al resto.
</p>

```

Resultado del ejemplo

Este párrafo, y sólo este párrafo, es verde, esta en negrita, tiene unos grandes márgenes y un borde azul.

Este párrafo es del color habitual, pero esta *palabra* es diferente al resto.

Combinando Hojas de Estilo

Se puede utilizar más de una hoja de estilo para fijar las características de un documento. Esto es deseable si se tienen varias hojas de estilos parciales, de forma que cada una de ellas define diferentes estilos. Supongamos, por ejemplo, que estamos escribiendo un informe sobre los beneficios de un producto de red de una compañía llamada RAD's. Puede que se necesite usar tres hojas de estilo: una definiendo el estilo habitual de los informes, otra que defina el estilo de los productos de red, y otra que defina el estilo de la compañía RAD's. El siguiente ejemplo muestra el uso de varias hojas de estilo en un mismo documento.

```

<style type="text/css"
    src="http://www.rads.com/estilo/empresa"
</style>
<style type="text/css"
    src="estilos/informe"
</style>
<style type="text/javascript"
    src="estilos/redes"
</style>

```

```
<style type="text/css"
  h1 {color: red;} " /* tiene preferencia sobre las hojas externas */
</style>
```

Entre las hojas de estilo externas, la última tiene precedencia sobre las demás. Así, en caso de conflicto, se escoge el estilo de la última hoja de estilo especificada. Los estilos definidos para elementos individuales tienen precedencia sobre los definidos en el elemento `<style>` y sobre los definidos en las hojas externas. En general, los estilos locales se sobreponen a los generales, como se muestra en el siguiente ejemplo.

Sintaxis CSS

```
<style type="text/css">
  p {color: white;}
  b {color: green;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  tags.p.color = "white";
  tags.b.color = "green";
</style>
```

Uso de la hoja de estilo

`<p>`Tal y como debería ser este párrafo es de color blanco, ``y esta parte en negrita es de color verde`</p>` `<p style="color: yellow;">`Este otro párrafo no es de color blanco, `<b style="color: red;">`ni esta parte en negrita es de color verde``, porque se ha usado `style` para cambiarles el color.`</p>`

Resultado del ejemplo

Tal y como debería ser este párrafo es de color blanco, **y esta parte en negrita es de color verde**

Este otro párrafo no es de color blanco, **ni esta parte en negrita es de color verde**, porque se ha usado `style` para cambiarles el color.

Propiedades de Formato de Elementos de Bloque

En esta sección se exponen las opciones de formato de los elementos de bloque. Los elementos de bloque comienzan en una nueva línea. Por ejemplo, `<h1>` y `<p>` son elementos de bloque, pero `` no lo es. Comenzaremos presentando unos ejemplos que muestren las posibilidades de formato de los elementos de bloque. Después se discutirá cada opción de formato en detalle. Para finalizar se echará un vistazo a su comportamiento respecto a la herencia de propiedades.

Formato de Bloques: Introducción y Ejemplos

Las hojas de estilo tratan a cada elemento de bloque como si estuviera rodeado de una caja. Cada caja puede tener características de estilo propias tales como márgenes, bordes, separadores y una imagen o color de fondo. Los márgenes indican la separación entre el borde de la caja y el borde del documento. Estos bordes pueden tener apariencia plana o tridimensional. Los separadores ("padding") indican la separación entre el borde de los elementos y el contenido de los mismos. También se puede fijar la anchura de los elementos de bloque, bien mediante un valor específico, o bien mediante un porcentaje de la anchura total del documento.

En este caso es redundante fijar los márgenes derecho o izquierdo y la anchura. Si se especifican la anchura y los dos márgenes, el valor del margen izquierdo tiene prioridad sobre los demás valores en caso de conflicto. En este caso el valor del margen derecho especifica la distancia máxima desde el borde derecho de elemento que lo contiene. El valor de anchura es utilizado sólo si no sobrepasa los límites de anchura del elemento que lo contiene. El alineamiento horizontal puede ser a la izquierda, derecha o centrado. Esto se consigue usando la propiedad [float](#) en [CSS](#) o la propiedad [align](#) en JavaScript. En los siguientes ejemplos se muestra el uso de márgenes, separadores, bordes, fondos y alineamiento.

Sintaxis CSS

```
<style type="text/css">
p {
  color: #ffffff; /* blanco */
  /* márgenes */
  margin-left: 20%; margin-right: 20%;
  /* anchura del borde */
  border-top-width: 10pt; border-bottom-width: 10pt;
  border-right-width: 5pt; border-left-width: 5pt;
  /* estilo y color del borde */
  border-style: outset; border-color: blue;
  /* separadores */
  padding-top: 10pt; padding-bottom: 10pt;
  padding-right: 20pt; padding-left: 20pt;
}
h3 {
  /* tamaño y peso de la fuente */
  font-size: 14pt; font-weight: bold;
  background-image: url("papel.jpg");
  /* centra el título y le da una anchura del 90% */
  width: 90%; float: center;
  borde-color: green; borde-style: solid;
  /* todas las partes del borde tienen la misma anchura */
  borde-width: 10pt;
  /* todos los separadores del borde son igual de anchos */
  padding: 5%;
}
```

```
}  
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">  
with (tags.p) {  
    /* márgenes */  
    marginLeft="20%"; marginRight="20%";  
    /* anchura del borde */  
    borderTopWidth="10pt"; borderBottomWidth="10pt";  
    borderRightWidth="5pt"; borderLeftWidth="5pt";  
    /* estilo y color del borde */  
    borderStyle="outset"; borderColor="blue";  
    /* separadores */  
    paddingTop="10pt"; paddingBottom="10pt";  
    paddingRight="20pt"; paddingLeft="20pt";  
}  
with(tags.h3) {  
    /* tamaño y peso de la fuente*/  
    fontSize="14pt"; fontWeight="bold";  
    backgroundImage="papel.jpg";  
    /* centra el titulo y le da una anchura del 90% */  
    width="90%"; float="center";  
    bordeColor="green"; bordeStyle="solid";  
    /* todas las partes del borde tienen la misma anchura */  
    bordeWidths("10pt");  
    /* todos los separadores del borde son igual de anchos */  
    paddings("5%");  
}  
</style>
```

Uso de la hoja de estilo

```
<h3>Titulo h3 con borde sólido y fondo</h3>  
<p>Los bordes se usan muy a menudo. Por ejemplo, si un bloque tiene borde resalta mucho más que si no lo tiene.</p>  
<p>Este es otro párrafo con borde. Ten cuidado con los bordes, no los hagas demasiado anchos, pues de lo contrario ocuparán demasiado espacio.</p>
```

Resultado del ejemplo

Titulo h3 con borde sólido y fondo

Los bordes se usan muy a menudo. Por ejemplo, si un bloque tiene borde resalta mucho más que si no lo tiene.

Este es otro párrafo con borde. Ten cuidado con los bordes, no los hagas demasiado anchos, pues de lo contrario ocuparán demasiado espacio.

Márgenes

Los márgenes indican la separación entre el borde del bloque y el borde del documento, o elemento padre. Se pueden fijar los márgenes derecho, izquierdo, superior e inferior. Para ello se deben utilizar los siguientes nombres de propiedad:

Sintaxis CSS

- `margin-top`
- `margin-bottom`
- `margin-left`
- `margin-right`
- `margin`

Sintaxis JavaScript

- `marginTop`
- `marginBottom`
- `marginLeft`
- `marginRight`
- `margins()`

En vez de especificar los dos márgenes se puede utilizar la propiedad `width`. Se pueden utilizar valores específicos, como 200 puntos, o valores relativos, como el 50% de la anchura del elemento padre.

Es redundante fijar los dos márgenes y la anchura, pues dos de estos tres valores implican el tercero. Para especificar los márgenes por defecto para un documento se deben especificar para la etiqueta `<body>`. En el siguiente ejemplo se fijan dichos márgenes en 20 puntos a derecha e izquierda.

Sintaxis CSS

```
<style type="text/css">
  body {margin-left: 20pt; margin-right: 20pt;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  with (tags.body) { marginLeft="20pt"; maginRight="20pt"; }
```

</style>

Bordes

Se puede fijar la anchura del borde que rodea un elemento de bloque usando las siguientes propiedades.

Sintaxis CSS

- `border-top-width`
- `border-bottom-width`
- `border-left-width`
- `border-right-width`
- `border-width`

Sintaxis JavaScript

- `borderTopWidth`
- `borderBottomWidth`
- `borderLeftWidth`
- `borderRightWidth`
- `borderWidths()`

Se puede fijar el estilo del borde usando la propiedad `border-style` de CSS o la propiedad `borderStyle` de JavaScript. Los valores que pueden tomar son



`solid`, `double`, `groove`, `ridge`, `inset` y `outset`.

Separadores

Los separadores indican la distancia entre el borde de un elemento y su contenido. El separador se muestra incluso si el borde del elemento no lo hace. Se puede fijar el tamaño de los separadores de un elemento de bloque utilizando las siguientes propiedades.

Sintaxis CSS

- `padding-top`
- `padding-bottom`
- `padding-left`
- `padding-right`

Sintaxis JavaScript

- `paddingTop`
- `paddingBottom`
- `paddingLeft`

- paddingRight

Herencia de Propiedades de Formato de Bloque

Las características de anchura, márgenes, bordes y separadores de los elementos padre no son heredadas por sus hijos. Sin embargo, a primera vista, a veces puede parecer que sí son heredados, pues los valores de los elementos padre afectan a sus elementos hijos. Supongamos que a un elemento `<div>` le fijamos un margen izquierdo de valor 10 puntos. De esta forma la caja que le rodea se encuentra desplazada 10 puntos hacia la derecha. Además supondremos que no tiene bordes ni separadores. Todos sus elementos hijo estarán pegados a su margen izquierdo, que como está 10 puntos desplazado hacia la derecha, causará un efecto parecido si ellos mismos también tuviesen borde izquierdo.

Pensemos en qué ocurriría si los hijos heredasen estas características de sus padres. El bloque `<div>` está indentado 10 puntos. Sus hijos a su vez estarán indentados otros 10 puntos con respecto a él, con lo cual el aspecto global es que los hijos estarían indentados 20 puntos.

Manual de Referencia de Hojas de Estilo

Comentarios en las Hojas de estilo

En las hojas de estilo en cascada y en las hojas de estilo en JavaScript se pueden usar comentarios al estilo C. Por ejemplo:

```
em {color:red;} /* el texto enfatizado será rojo */
tags.em.color="red"; /* el texto enfatizado será rojo */
```

JavaScript además permite usar comentarios al estilo de C++. Por ejemplo:

```
tags.em.color="red"; // el texto enfatizado será rojo
```

Los comentarios no pueden anidarse.

Nuevas etiquetas HTML

En esta sección se verán las nuevas etiquetas que han sido añadidas para trabajar con estilos.

<STYLE>

Las etiquetas `<STYLE>` y `</STYLE>` se usan para crear una hoja de estilo. En su interior podemos especificar estilos para elementos, definir clases e identificadores y en general establecer los estilos que se utilizarán en todo el documento. Para especificar qué tipo de sintaxis se empleará utilizaremos el atributo `TYPE`. Su valor por defecto es "text/css" y selecciona la sintaxis CSS. Mediante el valor "text/javascript" podemos seleccionar la sintaxis JavaScript.

Ejemplos:

Sintaxis CSS

```
<style type="text/css">
  body { margin-left: 10%; margin-right: 10%; }
  all.limon { color: yellow; }
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  tags.body.marginLeft="10%";
  tags.body.marginRight="10%";
  classes.limon.all.color="yellow";
</style>
```

<LINK>

La etiqueta `<LINK>` sirve para poder utilizar en un documento una hoja de estilo que está en otro fichero.

Sintaxis CSS

```
<html>
  <head>
    <title>Titulo</title>
    <link rel=stylesheet type="text/css" href="http://estilos/mi_estilo.html">
  </head>
  <body> ... </body>
</html>
```

Sintaxis JavaScript

```

<html>
  <head>
    <title>Titulo</title>
    <link rel=stylesheet type="text/javascript"
      href="http://estilos/mi_estilo.html">
  </head>
  <body> ... </body>
</html>

```


Las etiquetas y se utilizan para delimitar el principio y el fin de una trozo de código al que se le va aplicar un estilo. En el siguiente ejemplo se aplica un estilo a una parte del texto:

Este texto es normal. Este texto es diferente gracias a . De nuevo el texto es normal.

El resultado del ejemplo es:

Este texto es normal. Este texto es diferente gracias a . De nuevo el texto es normal.

Veamos otro ejemplo. Este hace diferentes las letras iniciales:

```

<style type="text/css">
  all.letron {font-family: times; font-size: 200%; font-weight: bold;}
</style>

<p><span class="letron">E</span>n un lugar de la Mancha...</p>

```

Resultado:

En un lugar de la Mancha...

Nuevos atributos para las etiquetas HTML

En esta sección se enumeran los nuevos atributos que podemos utilizar con las etiquetas HTML y que son útiles para trabajar con estilos. Estos atributos pueden ser usados con cualquier etiqueta para especificar su estilo.

STYLE

El atributo **STYLE** determina el estilo del elemento al que se aplique. Por ejemplo:

Sintaxis CSS

```
<h4 style="font-weight: bold; color: red;">Cabecera h4 en rojo</h4>
```

Sintaxis JavaScript

```
<h4 style="fontWeight='bold'; color='red'">Cabecera h4 en rojo</h4>
```

Resultado del ejemplo:

Cabecera h4 en rojo

CLASS

El atributo **CLASS** permite aplicar una clase de estilo a un elemento. Aunque CSS y JavaScript usen sintaxis ligeramente diferentes para la definición de clases de estilos, la forma de usar dichas clases es común. Ejemplo:

Sintaxis CSS

```
<style type="text/css">
  h3.cursiva { font-style: italic; }
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  classes.cursiva.h3.fontStyle="italic";
</style>
```

Uso de la hoja de estilo:

```
<h3 class="cursiva">Cabecera h3 cursiva</h3>
```

Se debe tener en cuenta que para los nombres de clase se distingue entre mayúsculas y minúsculas. Cada elemento HTML sólo puede utilizar una clase de estilo.

Para especificar que una clase puede aplicarse a todos los elementos se utiliza el selector `all` cuando se definen las propiedades de la clase. En el siguiente ejemplo todos los elementos de clase `limon` serán amarillos.

Sintaxis CSS

```
<style type="text/css">
  all.limon {color: yellow;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  classes.limon.all.color="yellow";
</style>
```

Uso de la hoja de estilo:

```
<h4 class="limon">Una cabecera amarilla</h4>
<p class="limon">Un párrafo amarillo</p>
```

Resultado del ejemplo

Una cabecera amarilla

Un párrafo amarillo

ID

Cuando se definen hojas de estilo, se pueden crear estilos individuales con nombre. Un elemento puede usar clases de estilos y además usar estos estilos individuales con nombre. Con ellos podemos crear excepciones a las clases de estilos para definir un estilo individual con nombre, en sintaxis CSS se utiliza el signo `#`, mientras que en JavaScript se utiliza el selector `ID`. En ambos casos se utiliza el atributo `ID` para especificar el estilo de un elemento HTML. En los nombres de los ID's se distingue entre mayúsculas y minúsculas. En el siguiente ejemplo se define la clase `calor`, que hace a los párrafos ser de color rojo y estar en negrita. También se crea un nombre de estilo, `frío`, que es de color azul. Se muestra cómo utilizar `frío` para crear una excepción a `calor`.

Sintaxis CSS

```
<style type="text/css">
  p.calor {color: red; font-weight: bold;}
  #frio {color: blue;}
</style>
```

Sintaxis JavaScript

```
<style type="text/javascript">
  classes.calor.p.color="red";
  classes.calor.p.fontWeight="bold";
  ids.frio.color="blue";
</style>
```

Uso de la hoja de estilo:

```
<p class="calor">Un párrafo caliente...</p>
<p class="calor" id="frio">Un párrafo frío...</p>
```

Resultado del ejemplo:

Un párrafo caliente...

Un párrafo frío...

Nuevas propiedades de objetos JavaScript

En esta sección se exponen las nuevas propiedades de los objetos JavaScript que son útiles para definir hojas de estilos usando la sintaxis de JavaScript.

TAGS

Cuando se usa sintaxis JavaScript dentro del elemento `<style>`, podemos seleccionar estilos usando la propiedad `tag` del objeto `document`.

El siguiente ejemplo utiliza la sintaxis JavaScript para especificar que todos los párrafos sean de color verde.

```
<style type="text/javascript">
  tags.p.color="green";
</style>
```

En sintaxis CSS este ejemplo seria:

```
<style type="text/css">
  p {color: green;}
</style>
```

La propiedad `tags` siempre se refiere al objeto `document` del documento actual, por esto se puede omitir `document` de la expresión `document.tags`. Así, las dos siguientes líneas expresan lo mismo:

```
document.tags.p.color="green";  
tags.p.color=green";
```

Para definir estilos por defecto a todos los elementos de un documento se debe hacer a través del elemento <body>, porque todos los demás elementos heredan de este.

```
tags.body.marginLeft="10%"; /* sintaxis JavaScript */  
body {margin-left: 10%;} /* sintaxis CSS */
```

CLASSES

Ver la sección CLASS de los nuevos atributos para las etiquetas HTML.

IDS

Ver la sección ID de los nuevos atributos para las etiquetas HTML.

Propiedades de las Hojas de Estilo

Fuentes

Tamaño de Fuente

Sintaxis CSS: `font-size`

Sintaxis JavaScript: `fontSize`

Posibles valores	tamaño absoluto, tamaño relativo, longitud, porcentaje
Valor inicial	<code>medium</code>
Se aplica a	todos los elementos
Heredable	si
Valores porcentuales	relativos al tamaño del padre

Tamaño absoluto: `xx-small`, `s-small`, `small`, `medium`, `large`, `x-large`, `xx-large`

Tamaño relativo: `larger`, `smaller`

Longitud: número + unidad

Porcentaje: tamaño de fuente relativo al tamaño de fuente del padre.

Tipo de Fuente

Sintaxis CSS: `font-family`

Sintaxis JavaScript: `fontFamily`

Posibles valores	un nombre de fuente
Valor inicial	la fuente por defecto
Se aplica a	todos los elementos
Heredable	si
Valores porcentuales	no aplicable

Tipo de fuente: `serif`, `sans-serif`, `cursive`, `monospace`, `fantasy`...

Peso de Fuente

Sintaxis CSS: `font-weight`

Sintaxis JavaScript: `fontWeight`

Posibles valores	<code>normal</code> , <code>bold</code> , <code>bolder</code> , <code>lighter</code> , 100 - 900
Valor inicial	<code>normal</code>
Se aplica a	todos los elementos
Heredable	si
Valores porcentuales	no aplicable

Estilo de Fuente

Sintaxis CSS: `font-style`

Sintaxis JavaScript: `fontStyle`

Posibles valores	<code>normal</code> , <code>italic</code>
Valor inicial	<code>Normal</code>
Se aplica a	todos los elementos
Heredable	si
Valores porcentuales	no aplicable

Propiedades del Texto

Interlinea

Sintaxis CSS: [line-height](#)
Sintaxis JavaScript: [lineHeight](#)

Posibles valores	número, longitud, porcentaje, normal
Valor inicial	tamaño por defecto para la fuente
Se aplica a	elementos de bloque
Heredable	Si
Valores porcentuales	relativos al tamaño de la fuente

Decoración del Texto

Sintaxis CSS: [text-decoration](#)
Sintaxis JavaScript: [textDecoration](#)

Posibles valores	none , underline , line-through , blink
Valor inicial	None
Se aplica a	Todos los elementos
Heredable	Si
Valores porcentuales	no aplicable

Transformaciones del Texto

Sintaxis CSS: [text-transform](#)
Sintaxis JavaScript: [textTransform](#)

Posibles valores	capitaliza , uppercase , lowercase , none
Valor inicial	none
Se aplica a	todos los elementos
Heredable	si
Valores porcentuales	no aplicable

Alineamiento del Texto

Sintaxis CSS: [text-align](#)
Sintaxis JavaScript: [textAlign](#)

Posibles valores	left , right , center , justify
Valor inicial	left
Se aplica a	Elementos de bloque
Heredable	Si
Valores porcentuales	no aplicable

Indentación de Texto

Sintaxis CSS: [text-indent](#)

Sintaxis JavaScript: [textIndent](#)

Posibles valores	longitud, porcentaje
Valor inicial	0
Se aplica a	elementos de bloque
Heredable	si
Valores porcentuales	relativos a la anchura del padre

Longitud: numero + unidad

Porcentaje: longitud de indentación relativo a la anchura del padre.

Propiedades de Formato de Elementos de Bloque

Márgenes

Sintaxis CSS: [margin-left](#), [margin-right](#), [margin-top](#), [margin-bottom](#), [margin](#)

Sintaxis JavaScript: [marginLeft](#), [marginRight](#), [marginTop](#), [marginBottom](#), [margins\(\)](#)

Posibles valores	longitud, porcentaje, auto
Valor inicial	0
Se aplica a	Todos los elementos
Heredable	No
Valores porcentuales	Relativos a la anchura del padre

Separadores

Sintaxis CSS: [padding-left](#), [padding-right](#), [padding-top](#), [padding-bottom](#), [padding](#)

Sintaxis JavaScript: [paddingLeft](#), [paddingRight](#), [paddingTop](#), [paddingBottom](#), [paddings\(\)](#)

Posibles valores	Longitud, porcentaje
Valor inicial	0
Se aplica a	Todos los elementos
Heredable	No
Valores porcentuales	Relativos a la anchura del padre

Anchura del los Bordes

Sintaxis CSS: `border-left-width`, `border-right-width`, `border-top-width`, `border-bottom-width`, `border-width`

Sintaxis JavaScript: `borderLeftWidth`, `borderRightWidth`, `borderTopWidth`, `borderBottomWidth`, `borderWidths()`

Posibles valores	Longitud
Valor inicial	None
Se aplica a	Todos los elementos
Heredable	No
Valores porcentuales	No aplicable

Estilo del Borde

Sintaxis CSS: `border-style`

Sintaxis JavaScript: `borderStyle`

Posibles valores	None, solid, double, inset, outset, groove, ridge
Valor inicial	None
Se aplica a	Todos los elementos
Heredable	No
Valores porcentuales	No aplicable

Color del Borde

Sintaxis CSS: `border-color`

Sintaxis JavaScript: `borderColor`

Posibles valores	Color, none
------------------	-------------

Valor inicial	None
Se aplica a	Todos los elementos
Heredable	No
Valores porcentuales	no aplicable

Color: nombre de un color o 6 dígitos hexadecimales indicando su valor rgb.

Anchura

Sintaxis CSS: `width`

Sintaxis JavaScript: `width`

Posibles valores	longitud, porcentaje, <code>auto</code>
Valor inicial	<code>auto</code>
Se aplica a	elementos de bloque
Heredable	no
Valores porcentuales	relativos a la anchura del padre

Alineamiento

Sintaxis CSS: `float`

Sintaxis JavaScript: `align`

Posibles valores	<code>left</code> , <code>right</code> , <code>none</code>
Valor inicial	<code>none</code>
Se aplica a	todos los elementos
Heredable	no
Valores porcentuales	relativos a la anchura del padre

Clear

Sintaxis CSS: `clear`

Sintaxis JavaScript: `clear`

Posibles valores	<code>none</code> , <code>left</code> , <code>right</code> , <code>both</code>
Valor inicial	<code>none</code>

Se aplica a	Todos los elementos
Heredable	No
Valores porcentuales	no aplicable

Esta propiedad especifica cuando un elemento permite la existencia de elementos flotando a sus lados. Exactamente, el valor de esta propiedad son los lados en que no se aceptan elementos flotantes.

Propiedades de Color y Fondo

Color

Sintaxis CSS: `color`

Sintaxis JavaScript: `color`

Posibles valores	<code>Color</code>
Valor inicial	<code>Black</code>
Se aplica a	Todos los elementos
Heredable	Si
Valores porcentuales	no aplicable

Esta propiedad especifica el color del texto.

Color: especificación de un color (nombre, valor rgb).

Imagen de Fondo

Sintaxis CSS: `background-image`

Sintaxis JavaScript: `backgroundImage`

Posibles valores	<code>url</code>
Valor inicial	Vacio
Se aplica a	todos los elementos
Heredable	no
Valores porcentuales	no aplicable

Color de Fondo

Sintaxis CSS: `background-color`

Sintaxis JavaScript: `backgroundColor`

Posibles valores	<code>color</code>
Valor inicial	vacio
Se aplica a	todos los elementos

Heredable	no
Valores porcentuales	no aplicable

Propiedades de Clasificación

Display

Sintaxis CSS: [display](#)

Sintaxis JavaScript: [display](#)

Posibles valores	ninguno, block , inline , list-item
Valor inicial	según HTML
Se aplica a	todos los elementos
Heredable	no
Valores porcentuales	no aplicable

Esta propiedad indica cuando un elemento es en línea, como ``, de bloque, como `<h1>` o de lista, como ``.

Estilo de Lista

Sintaxis CSS: [list-style-type](#)

Sintaxis JavaScript: [listStyleType](#)

Posibles valores	ninguno, disk , circle , square , decimal , lower-roman , upper-roman , lower-alpha , upper-alpha
Valor inicial	Disk
Se aplica a	elementos cuya propiedad display es list-item
Heredable	Si
Valores porcentuales	no aplicable

Espacios en Blanco

Sintaxis CSS: [white-space](#)

Sintaxis JavaScript: [whiteSpace](#)

Posibles valores	normal , pre
Valor inicial	según HTML
Se aplica a	elementos de bloque
Heredable	Si

Valores porcentuales	no aplicable
----------------------	--------------

Unidades

Unidades de Longitud

El formato de un valor de longitud es un signo opcional ('+' o '-', '+' por defecto), un número y una unidad de medida. Ejemplos: 12pt, 2em, 3mm.

Hay tres tipos de unidades: absolutas, relativas y pixel.

Tipos de unidades absolutas:

- pt: puntos
- pc: picas
- px: pixels
- in: pulgadas
- mm: milímetros
- cm: centímetros

Tipos de unidades relativas:

- em: la altura de la fuente, normalmente la anchura o altura de la M mayúscula
- ex: la mitad de la altura la fuente, normalmente la altura de la letra 'x'
- px: pixels, relativos a la superficie de dibujo

Los elementos hijo heredan los valores calculados y no los valores relativos. Por ejemplo:

```
body {font-size: 12; text-indent: 3em}
h1 {font-size: 15pt;}
```

En este ejemplo la indentación del texto en <h1> será 36pt y no 45pt.

Unidades de Color

Un valor de color es o un nombre de color o una descripción RGB.

La lista de colores sugerida es: [aqua](#), [black](#), [blue](#), [fuchsia](#), [gray](#), [green](#), [maroon](#), [navy](#), [olive](#), [purple](#), [red](#), [silver](#), [teal](#), [white](#), [yellow](#). Estos 15 colores se han tomado de la paleta VGA de Windows.

```
body {color: black;}
tags.body.backgroundColor = "white"
```

Un color RGB puede especificarse mediante 6 dígitos hexadecimales. Los dos primeros indican rojo, los dos segundos verde y los dos últimos azul. Ejemplos:

```
body {color: #ff0000;} /* rojo */  
tags.p.backgroundColor = "#606060"; /* gris */
```

También podemos utilizar la función rgb(). Usa tres argumentos: rojo, verde y azul. Cada color puede ser un entero entre 0 y 255, o un porcentaje. Ejemplo:

```
p {color: rgb(200, 20, 240);} /* púrpura brillante */  
blockquote {background-color: rgb(100%, 100%, 20%);} /* amarillo brillante */
```

Creación de **David De Belaustegui**
Para **SOFTDOWNLOAD ARGENTINA**
<http://www.softdownload.com.ar>