II Curso Online JAVA-J2EE

TEMA 2

Capa de presentación en entornos Web

CENTRO DE EXCELENCIA
DE SOFTWARE LIBRE

CASTILLA-LA MANCHA

Autor: PCYTA / Centro de Excelencia de Software Libre de Castilla-La Mancha
Versión: 1.0
Fecha: Revisado 13-02-2008 23:24

0 Licencia

UNIDAD I. Capa de presentación en entornos Web

1 HTML

1.1 ¿Qué es HTML?

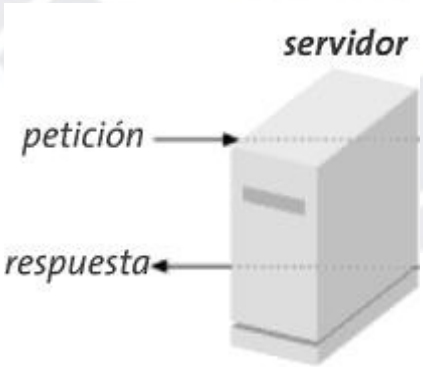
Antes de entrar a ver el concepto HTML es interesante tener algunas nociones de lo que es el protocolo HTTP.

HTTP (HyperText Transfer Protocol) es un protocolo de red usado para el envío de datos y ficheros en una red como por ejemplo Internet. Este protocolo funciona sobre TCP/IP.

HTTP es un protocolo sin estado. Esto quiere decir que cada petición debe contener todo lo necesario para que el servidor construya la respuesta.

Lo que conocemos como navegadores (FireFox, Internet Explorer, Opera,...) son, entre otras funcionalidades, **clientes HTTP**. Estos navegadores realizan peticiones HTTP (**request**) a un servidor Web (Apache Web Server, IIS,...) y los servidores responden a esa petición devolviendo una respuesta (**response**) al cliente. Este cliente recibe la respuesta y ejecuta las acciones necesarias para, por ejemplo, mostrársela al usuario.

El puerto estándar (en el que escuchan los servidores Web) para el protocolo HTTP es el 80 aunque se puede usar cualquier otro.



HTML (HyperText Markup Language) es un conjunto de etiquetas incluidas en archivos de texto que definen la estructura de una página Web y sus vínculos a otros recursos. Los navegadores leen estos archivos de texto e interpretan esas etiquetas para determinar como mostrar la página Web. Un fichero HTML debe tener como extensión htm o html, y como fichero de texto que es, se puede

crear con un simple editor de texto (notepad, vi,...)

Una aclaración interesante es diferenciar entre página Web y fichero HTML. Una página Web es el compendio del fichero HTML y todos los recursos asociados, como imágenes, ficheros de recursos JavaScript, ficheros de definición de estilos, etc.

1.2 Estructura básica de un fichero HTML

Un fichero **HTML** esta compuesto por 2 partes: el encabezamiento (head) y el cuerpo (body) de la página. Esto, en etiquetas html se traduce en algo similar a lo siguiente:

```
<html>
<head>
        Encabezamiento de la página
</head>
<body>
        Cuerpo del fichero HTML
</body>
</html>
```

Esta estructura es recomendable y prácticamente **obligatoria**. Decimos que es prácticamente obligatoria porque los navegadores son capaces de interpretar ficheros html que contengan, por ejemplo, etiquetas mal cerradas; y decimos que es recomendable porque si el fichero no está bien formado, podemos llevar a la confusión al navegador.

Por ejemplo, si insertamos el contenido anterior en un fichero con extensión .html y lo abrimos con un navegador, se mostrará en la pantalla el texto *Cuerpo del fichero HTML* .

Además de estas tres etiquetas, existe un conjunto mucho más numeroso que usadas de la forma apropiada y unidas a los recursos necesarios darán forma a la página Web que queremos mostrar. Esas etiquetas son mostradas en el siguiente apartado.

1.3 Etiquetas básicas de HTML

En HTML existe un numeroso conjunto de etiquetas para muy diversas finalidades. Tenemos etiquetas para formateo de texto, para asignar colores a regiones, para ordenar listas de datos, para definir los datos a ser enviados al servidor, etc. Muchas etiquetas pueden ser ampliadas mediante el **uso de atributos**. A continuación detallamos las etiquetas básicas y ejemplos autoexplicativos:

- **Encabezados** (Headings) para definir diferentes niveles de título. Van de *<h1>* hasta *<h6>* donde *<h1>* es el de mayor tamaño y *<h6>* el de menor. Como ejemplo está lo siguiente:

```
<h1>Título del tema</h1>
<h2>Apartado del tema</h2>
<h3>Sub-apartado</h3>
& & & & ..
<h6>Nivel más pequeño</h6>
```

  Antes y después de cada encabezado se inserta automáticamente un retorno de carro.

- **Párrafos** para definir que un texto es un párrafo. Para esto se utiliza la etiqueta *<p>*. Ejemplo:

```
<p>Esto es un párrafo</p>
<p>Esto es otro párrafo</p>
```

  Al igual que con los encabezados, se inserta de forma automática antes y después un retorno de carro.

- Carácter de **nueva línea**. Con la etiqueta *
* intercalada en el texto hacemos que el navegador muestre el resto del texto en la siguiente linea. Por ejemplo:

```
Línea 1<br>Línea 2
```

  Cabe destacar el detalle que la etiqueta <br> no tiene cierre.

- **Comentarios** en HTML. Todo el texto intercalado entre <!-- y --> es un comentario. Por

ejemplo:

```
<!-- comentario -->
```

- Etiquetas para dar **formato al texto**. A continuación mostramos ejemplos de las etiquetas más importantes:

```
<b>negrita</b>
<big>texto grande</big>
<em>texto enfatizado</em>
<i>cursive</i>
<small>texto pequeño</small>
<strong>texto enorme</strong>
<sub>subíndice</sub>
<sup>superíndice</sup>
```

- **Enlaces** a otras páginas:

```
<a href= url >Texto mostrado</a>
<a href="http://www.iformacion.es">El mejor sitio de eLearning</a>
```

- **Frames**. El uso de frames permite mostrar varios documentos HTML en el mismo navegador. Un ejemplo típico es dividir la página Web a mostrar en dos frames, poniendo en uno de ellos el menú de la aplicación y en otro el contenido del enlace seleccionado en el menú. A pesar de ser usado, **no** es muy recomendado por la complejidad que impone al desarrollador el control de los mismos. Además, en posteriores versiones de la especificación HTML se han incluido nuevas etiquetas que permiten esta funcionalidad evitando gran parte de las desventajas.

Un ejemplo típico de uso es el siguiente. En él que se puede ver como se separa la página Web en dos partes, en la izquierda se encontraría el menú y en la derecha el contenido de la página.

```
<frameset cols="20%,80%">
    <frame src="menu.html">
```

```
        <frame
        src="contenido.htm">
</frameset>
```

- **Tablas**. Este elemento es muy utilizado tanto para mostrar datos como para dar forma a páginas Web. Una tabla (<*table*>) se compone de filas (<*tr*>) y celdas dentro de las filas (<*td*> ó <*th*> si es es cabecera). Una <*td*> o <*th*> puede contener texto, imágenes, otras tablas, etc. A continuación se muestra un ejemplo de tabla:

```
<table border=1>
        <tr>
                <th
                colspan="2">Abreviatu
                ras</th>
</tr>
        <tr>
                <td>pts</td>
                <td>pesetas</td>
</tr>
<tr>
                <td>etc</td>
                <td>etcétera</td>
</tr>
</table>
```

Como se puede observar, en este ejemplo se han utilizado los atributos que antes al principio del apartado comentábamos. En este caso se ha añadido el atributo *border=1* a la etiqueta *table*, consiguiendo que el borde de la tabla se a de un píxel. También hemos utilizado el atributo *colspan=2*, que indica que la celda ocupa el tamaño de dos, haciendo que la celda de la cabecera tenga la misma anchura que las dos celdas que componen el resto de filas.

- **Listas**. Existen tanto **numeradas** (<*ol*>) como **no numeradas** (<*ul*>). A continuación se muestran dos ejemplos explicativos:

Código HTML	Resultado
** ** Casa	1 Casa 2 Perro

Perro ** **	
** ** - Casa - Perro ** **	* Casa * Perro

Por supuesto, dentro de cada elemento <*li*> pueden introducirse texto, imágenes, tablas, etc.

- **Formularios**. Éstos son utilizados para el **envío de datos** a, por ejemplo, otra página Web. Los formularios (<*form*>) contienen elementos de formulario que serán los datos a ser enviados. Estos elementos pueden ser campos de texto, menus, checkbox, radio buttons, etc. A continuación se muestra un ejemplo de formulario:

```
<form name="miformulario" action="paginaQueRecibeDatos">
        First name: <input type="text" name="nombre"><br>
        Last name: <input type="text" name="apellido"><br>
        <input type="radio" name="sexo" value="hombre"> Hombre<br>
        <input type="radio" name="sexo" value="mujer"> Mujer<br>
        Me gusta:
        Leer
        <input type="checkbox" name="aficiones" value="lectura"/><br/>
        Hacer deporte
        <input type="checkbox" name=" aficiones " value="deporte"/><br/>
        Ir de bares <input type="checkbox" name=" aficiones "
        value="bares"/>
        <br>
        <input type="submit" value="enviar datos" />
</form>
```

Como se puede ver en el ejemplo, disponemos de un formulario llamado *miformulario* que manda una serie de datos a la página *paginaQueRecibeDatos* . Esos datos son el nombre, apellidos, sexo, y ciertas preferencias sobre actividades de ocio. Por último, se tiene un boton para enviar el formulario. Existen además otros tipos de campos de entrada de datos en un formulario, que permiten meter textos largos, ficheros, campos ocultos, contraseñas, listas de elementos seleccionables, etc. Para ello consultar la siguiente referencia.
http://www.htmlquick.com/es/reference/tags/input.html

1.4 Otras características de HTML

- **Caracteres especiales**. Un detalle importante es cuando se quieren introducir en una página caracteres que son reservados por el lenguaje de marcas (*HTML*). Por ejemplo, para introducir el símbolo < se debe escribir < . A continuación se muestra una tabla con algunos de los más reseñables:

Carácter	Código HTML
espacio	
<	<
>	>
&	&
"	"

- **Colores**. Por otro lado, la definición de colores en html sigue el estandar **RGB** con valores de 0 a 255 para cada **componente**. Aunque también existen ciertos **valores predefinidos** para algunos colores. Por ejemplo, para poner en color negro el fondo de la página Web, se pueden utilizar estas tres expresiones:

```
<body bgcolor="#000000">
<body bgcolor="rgb(0,0,0)">
<body bgcolor="black">
```

2 JAVASCRIPT

2.1 ¿Qué es JavaScript?

JavaScript es un lenguaje de scripting que fue diseñado para añadir interactividad a las páginas HTML. JavaScript es un lenguaje interpretado y poco pesado. Además, tiene una sintaxis muy simple.

Con JavaScript podemos escribir las acciones a realizar en una página Web al producirse **eventos** como pinchar algo, pasar con el ratón por encima, etc.

JavaScript nos **permite modificar el código HTML** de una página Web, permitiendo modificar el aspecto de una página. También nos ofrece la posibilidad de **leer propiedades del navegador** que visualiza la página HTML.

2.2 Uso de JavaScript en páginas WEB

Para incluir JavaScript en una página HTML basta con meter código JavaScript dentro de un par de etiquetas <script>. Seguidamente se muestra un ejemplo:

```
<html>

        <body>
                <script type="text/javascript">
                        alert("Hola Mundo")
</script>
</body>

</html>
```

Con el código del ejemplo, al cargarse la página se mostraría un mensaje con el texto Hola Mundo y un botón Aceptar para cerrar ese mensaje.

El código **JavaScript se ejecuta inmediatamente después de que la página se cargue en el**

navegador. Por otro lado, y quizás su funcionalidad más interesante, se puede **ejecutar código JavaScript como respuesta a eventos** sucedidos en la página.

El sitio ideal para definir el código JavaScript es en el aparatado <head> ya que así nos aseguramos de que siempre estará cargado. Aunque realmente, lo habitual y recomendable es definir el código **JavaScript en ficheros de extension *js* y declararlo en el apartado <head>** del fichero HTML. A continuación mostramos un ejemplo de lo dicho anteriormente:

```
<html>

        <head>
                <script
                src="miCodigoJavaScript.js"></script>
</head>
<body>
                .....
</body>
</html>
```

Con el código del ejemplo, conseguimos que se ejecute el fichero *miCodigoJavaScript.js* justo después de cargarse la página.

A pesar de todo esto, el uso más habitual de JavaScript consiste en definir funciones en código JavaScript que serán ejecutadas como respuesta a los distintos eventos.

**A lo largo del tema veremos ejemplos de este uso.*

2.3 Programación con JavaScript. Elementos Básicos

En este apartado vamos a hacer una revisión mediante ejemplos de los elementos básicos que se emplean para escribir código en JavaScript.

Elemento	Ejemplo	Comentario
Variables	**var** miVariable = Hola Mundo ;	Las variables se definen con la palabra ***var*** (aunque puede ser obviada).

	miVariable2=58;	No se definen tipos, una variable puede contener una cadena de texto, un valor numérico, un objeto, etc.

Condicionales	var fecha = new Date()
	var hora = fecha.getHours()
	if (hora<11) {
	document.write("Buenos días")
	} **else if** (hora >11 && hora <20) {
	document.write("Buenas tardes")
	} **else** {
	document.write("Buenas Noches")
	}	

Switch	**switch** (theDay)
	{
	case 5:
	document.write("Finally Friday")
	break
	case 6:
	document.write("Super Saturday")
	break
	case 0:
	document.write("Sleepy Sunday")
	break
	default:
	document.write("I'm looking forward to this weekend!")
	}
		Si no se pone el break pasaría al siguiente caso automáticamente.

Operadores	x=5; y=2; *x+y*; //7	Como se puede observar, es una sintaxis igual a la de Java. Resulta interesante el

	x-y; //3
	*x*y*; //10
	x/y; //2.5
	x%y; //1(módulo)
	x++; //6
	x--; //4
	x==*y*; //false
	5== *5* ; //true
	5=== *5* ; //false (igual valor y tipo);
	x!=*y*; //true
	x>*y*; //true
	x<*y*; //false
	x>=*y*; //true
	x<=*y*; //false
	true&&false; //false
	true||*false*; //true;
	!true; //false
		operador '===' que devuelve *true* si ambos operandos son iguales tanto en valor como en tipo.

Popups	**alert**("Hola Mundo") **confirm**("¿Está seguro?") **prompt**("Nombre","valorPor Defecto")	- **alert**muestra un popup con el mensaje y el botón aceptar . - **confirm** muestra un popup con el mensaje y el botón aceptar y cancelar . Si pulsas aceptar la función devolverá true y si pulsas cancelar la función devolverá false . - **prompt** solicita un valor al usuario y lo devuelve como resultado.

Funciones	**function** sumarMostrarYDevolver(a,b)
	{
	var resultado = a + b;
	alert("La suma es: " + resultado);
	return resultado;
	}
		Se puede observar que no se especifica en ningún momento el tipo de las variables y el tipo devuelto por una función. Además, tampoco es obligatorio devolver un resultado, con lo que no es obligatoria la sentencia return ;

Bucles For	var contador; **for**(contador=0;contador<=5;	

	contador++){
	alert("Contador = " +
	contador);
	}	

Bucles While	var contador=0;
	while (contador<=5) {
	alert("Contador = " +
	contador);
	contador++;
	}	

Todos estos elementos definen los aspectos más importantes de la sintaxis de JavaScript. Como podemos observar en lo visto hasta el momento, estamos viendo JavaScript desde el punto de vista del paradigma de programación **imperativo**. **JavaScript también está preparado para soportar el paradigma orientado a objetos**, como veremos más adelante.

Por otro lado, además de los elementos mostrados anteriormente, existen otros aspectos que veremos en el siguiente apartado.

2.4 Otros aspectos de Javascript

En este apartado vamos a explicar ciertos conceptos muy interesantes para usar JavaScript. Estos aspectos son la vinculación de código JavaScript a eventos en las páginas, gestión de excepciones y presentación de caracteres especiales.

- Eventos: Al escribir el código HTML de una página Web, se pueden vincular a los distintos elementos HTML cierto código JavaScript, el cual será ejecutado como respuesta a esos eventos. Seguidamente vemos un ejemplo de esta temática:

```
<input type="text" id="fecha" onchange="comprobarFormatoFecha()">
```

Como se puede ver en el ejemplo, existe un campo de texto que contendrá una fecha, y cuando este campo es modificado (***onchange***) se llama a una función JavaScript que comprobará el formato del mismo, pudiendo por ejemplo notificar al usuario de que la ha introducido mal.
Cada elemento HTML tiene sus eventos asociados, a continuación detallamos los principales

eventos que se usan en las páginas Web:

Evento	Comentario
onblur	Cuando el elemento pierde el foco
onchange	Cuando el usuario cambia el contenido del campo
onclick	Cuando el elemento HTML es clicado
onfocus	Cuando el elemento obtiene el foco
onkeydown	Cuando una tecla es presionada
onload	Cuando el elemento es cargado completamente
onmouseover	Cuando el ratón pasa sobre el elemento
onsubmit	Cuando el formulario va a ser enviado
onunload	Cuando el usuario sale de la página

Nota 1: existen más eventos referentes a ratón y teclado, cambios en el navegador o en el formulario, en respuesta a errores, etc. Pero con los vistos hasta el momento es suficiente para los objetivos del curso.
Nota 2: hay que tener en cuenta que no todos los navegadores aceptan todos los eventos en todas sus versiones. Y además, que no todos los eventos son lanzados en todos los elementos.

- **Excepciones** en JavaScript: En JavaScript se utilizan excepciones, siendo el concepto similar al que ya conocemos del lenguaje de programación JAVA. A continuación mostramos un pequeño código de ejemplo que utiliza las excepciones.

```
<script type="text/javascript">
var valorIntroducido=prompt("Inserta un número enter 1 y 5:","")
try {
        if(valorIntroducido >5)
                throw "mayor"
else if(valorIntroducido<1)
                throw "menor"
} catch(er) {
        if(er=="mayor")
                alert("El valor es mayor que 5")
if(er == "menor")
                alert("El valor es menor que 1")
```

```
}
</script>
```

- Introducción de **caracteres especiales**: Al igual que en Java, existen cierto caracteres especiales que para ser insertados en una cadena de texto deben ser escapados , es decir, poner el símbolo \ delante. A continuación vemos algunos ejemplos de estos caracteres:

Caracter	Símbolo
Comilla simple	\'
Doble comilla	\"
&	\&
Barra invertida	\\
Nueva línea	\n
Retorno de carro	\r
Tabulador	\t

2.5 Objetos en JavaScript

Con JavaScript también existe la posibilidad de definir objetos para conseguir las bondades de la programación orientada a objetos (POO). Esto puede ser bastante útil para nuestros desarrollos pero escapa de los objetivos de este curso. Además de poder definir tus propios objetos, JavaScript ya tiene definidos ciertos objetos que nos facilitan la programación imperativa que hemos visto hasta el momento. En la siguiente tabla vemos unos cuantos ejemplo del uso de estos objetos:

Objeto	Ejemplo	Comentarios
String	var texto = hola mundo ; alert(texto.**toUpperCase()**);	Como se puede ver, la variable texto es un objeto de tipo String del que se puede utilizar la función toUpperCase() para convertirla en mayúsculas. Además, String tiene otra serie de funciones y propiedades que permiten trabajar de forma más cómoda y pontente sobre cadenas de texto. Ver, por ejemplo,

		http://www.w3schools.com/jsref/jsref_obj_string.asp

Date	var miFecha=new **Date()**
	miFecha.**setFullYear**(2008,0,1)
	var hoy = new **Date**()
	if (miFecha>hoy)
	alert("antes de 1-1-2008")
	else
	alert("despues de 1-1-2008")
		Esta clase nos premite trabajar con fechas en JavaScript. Para ver en detalle las funciones y variables que ofrece mirar http://www.w3schools.com/jsref/jsref_obj_date.asp />

Arrays	var colores=new **Array()** colores[0]="Rojo" colores[1]="Azul" colores[2]="Verde"	Un array en JavaScript en un conjunto ordenado de elementos, los cuales pueden ser de cualquier tipo. Para ver todas las funciones y variables que ofrecen los Arrays consultar, por ejemplo, http://www.w3schools.com/jsref/jsref_obj_array.asp
Math	/*devuelve un número aleatorio*/ **Math.random()** /*devuelve el máximo de los valores x e y*/ **Math.max(x,y)** /*devuelve el valor absoluto de x*/ **Math.abs(x)**	Aquí vemos algunos ejemplos de las funciones que ofrece la clase Math. Para ver una completa especificación de esta clase, ver http://www.w3schools.com/jsref/jsref_obj_math.asp

2.6 Comentarios sobre JavaScript

Además de lo visto hasta ahora, en JavaScript se permiten muchas otras posiblidadades. Por ejemplo, se pueden manipular propiedades del navegador, cookies, cambiar estilos de los elementos HTML, modificar el árbol DOM de una página Web, etc. Cosas que no tocaremos en este curso por sobrepasar los objetivos del mismo.

Finalmente, y a modo de reflexión, es interesante conocer ciertos aspectos sobre el desarrollo con JavaScript:

- Es un código que se ejecuta en el navegador del cliente de modo que es muy difícil de controlar por parte del desarrollador. Existen multitud de navegadores, y también muchas versiones distintas

de los mismos, de modo que nuestro código JavaScript se puede comportar de formas distintas.

- En ocasiones, ciertos navegadores lo pueden tener deshabilitado, o incluso no disponer de la funcionalidad necesaria para ejecutarlo.

- La depuración en JavaScript es bastante compleja, aunque existen plugins instalables en los navegadores para facilitar esta labor.

- Existen también bastantes recursos que facilitan el desarrollo con JavaScript. Suelen ser librerías de funciones JavaScript que implementan funcionalidades muy comunes y que nos ahorran bastante trabajo, a la par que aumentan la calidad de nuestras aplicaciones. Véanse por ejemplo *RICO* - http://openrico.org o *DOJO* - http://www.dojotoolkit.org

3 CSS (Cascade Style Sheets)

3.1 ¿Qué es CSS?

Primero hagamos un ejercicio de visión global. **HTML**, como hemos visto anteriormente, **se pensó inicialmente para definir el contenido de un documento. Pero no se pensó demasiado en el estilo visual**. Para conseguir este aspecto visual, las empresas interesadas en ellos se encargaban de definir pequeñas ampliaciones de HTML incluyendo nuevas etiquetas para conseguir definir el aspecto gráfico.

Para estandarizar una especificación que tuviera en cuenta estos cambios, la **W3C creó los styles añadiéndolos a la especificación 4.0 de HTML**.

Un style es simplemente una definición de cómo se debe mostrar un elemento HTML. Más adelante veremos algún ejemplo pero valga decir que un estilo puede definir, por ejemplo, que un elemento table se muestre con un borde de 1 píxel, con un color de fondo verde para la cabecera y fondo azul el resto de celdas.

Los styles se agrupan en style sheets, de modo que finalmente en un fichero (representación física de un style sheet) se encontrarán todos los estilos definidos para un fichero HTML.

Una práctica muy habitual, y recomendable, es **definir todos los estilos necesarios para nuestra página Web en un fichero de extensión css y asociarlo a nuestra página Web** consiguiendo así que si queremos cambiar el estilo de nuestra página Web únicamente tendremos que tocar ese fichero de extensión css .

Una de las principales ventajas de las hojas de estilos es que **ahorran mucho trabajo a la hora de desarrollar y mantener una página Web**, y más aún si se trata de un portal compuesto de multitud de páginas.

El concepto **Cascading Style Sheets viene de la forma en la que, si varios estilos son aplicables a un elemento HTML, el estilo resultante se calcula en forma de cascada**, como si se sumaran todos los estilos aplicables. Más adelante veremos un ejemplo que dejará claro el concepto.

3.2 Definición de Styles para un fichero HTML

En este apartado vamos a ver como definir Styles para dar el aspecto visual a un fichero HTML. La sintaxis para la definición de un estilo es:

```
identificador {
        propiedad1:
        valor1;
        propiedad2:
        valor2
}
```

donde:
- ***identificador*** es el elemento al que queremos aplicar el estilo. También puede ser el nombre del estilo. Más adelante veremos en los ejemplos estos dos usos.
- ***propiedad*** es el atributo al que queremos asignar el valor.
- ***valor*** es el valor que se le asigna a la propiedad.

A continuación mostramos unos cuantos ejemplos:

Estilo	Comentario

body {
backgroun
d-color:
green;
}
	indica que el fondo del *body* será de color verde.

p {
text-align:
center;
color:
#FFFFFF;
font-
family:
"sans
serif"
}
	indica que los párrafos serán con texto centrado, en negro (FFFFFF) y con el tipo de letra sans serif *Nótese que si el valor son varias palabras se deben poner entre comillas.*

h1,h2,h3 {
color:
green;
}
	indica que los encabezados del nivel 1 al 3 irán en verde.

p.miEstilo {
text-align:
right;
color=gree
n
}
	indica que todos los elementos HTML de tipo que tengan el atributo class= miEstilo tendrán el texto a la derecha y el color verde.
	* ... *

.miEstilo2 {
text-align:
center;
border=1
}
	indica que todos los elementos HTML que tengan como atributo class= miEstilo2 tendrán el texto centrado y el borde de 1 pixel.
	* ... *

#miIdent {
color:
green
}
	cualquier elemento con id= miIdent se mostrará en verde.
	* ... *

p#miIdent2
{
text-align:
center;
color: red
}
	los elementos HTML de tipo con id= miIdent2 se mostrarán en rojo y con texto centrado
	* ... *

En CSS, el **código comentado empieza por /* y acaba por */**.
En el siguiente apartado veremos las diferentes formas de meter una definición de estilos en un fichero HTML.

3.3 Cómo insertar una hoja de estilos

Existen tres **formas de inserta los estilos** a emplear en una página Web.
1. Mediante un **fichero independiente**: Es la **forma más recomendable y usada** sobre todo cuando

el número de ficheros de la página Web es grande. Consiste en tener todos los estilos definidos en un fichero de extensión css y luego referenciarla en los ficheros HTML. A continuación se muestra un ejemplo de este método donde misEstilos.css es el fichero donde definimos nuestros estilos.

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

Cuando el navegador lea la etiqueta <link> automáticamente cargará los estilos del fichero y dará formato al documento HTML.

Un ejemplo muy simple del fichero misEstilos.css sería algo como esto:

```
p {
        margin-left: 20px
}
body {
        background-image:
        url("imagenDeFondo.jpg")
}
```

2. **En la cabecera del documento HTML**. Esta forma es **útil cuando el estilo es únicamente utilizado en una página HTML**. Seguidamente vemos un ejemplo de uso:

```
<head>
        <style type="text/css">
                p {margin-left: 20px}
                body {background-image:
                url("imagenDeFondo.jpg")}
</style>
</head>
```

3. **Estilo directo**. Se utiliza **para definir el estilo de un elemento concreto**. Solo es recomendable usarlo **cuando el estilo se va a asignar a un único elemento**. Un ejemplo sería el siguiente:

```
<p style=" margin-left: 20px">texto</p>
```

Siempre definir los estilos en un fichero css ya que **maximiza** la **independencia** entre datos y presentación de

los mismos; y además, también **maximiza** la reutilización de los estilos.

3.4 Trabajando con multiples Style Sheets

Como hemos visto hasta ahora, existen varias formas de definir los estilos, y también existe la posibilidad de que se haga referencia a varios ficheros *css* en un fichero HTML (definiendo varios elementos *LINK*), de modo que haya **diferentes definiciones para un mismo estilo**. Veamos un ejemplo que lo ilustra:

Fichero **misEstilos1.css**

```
h1 {
        color:blue;
        text-align: left;
}
```

Fichero **misEstilos2.css**

```
h1 {
        font-size: 10pt;
        text-align: right;
}
```

Fichero **miPagina.html**

```
<html>
<head>
        <link rel="stylesheet" type="text/css"
        href="misEstilos1.css" />
        <link rel="stylesheet" type="text/css"
        href="misEstilos2.css" />
</head>
<body>
        <h1>Esta es mi página</h1>
```

```
</body>
</html>
```

Con los tres ficheros de la definición anterior, el navegador cargaría los estilos del fichero *misEstilos1.css*, posteriormente cargaría los del fichero *misEstilos2.css***machacando las propiedades coincidentes** con las ya existentes (en este caso las del primer fichero) **y añadiendo el resto** a la definición global de estilos. De esta forma, se obtendría el siguiente resultado:

Definición global

```
h1 {
        color:blue;
        text-align: right;
        font-size: 10pt
}
```

De modo que el navegador pintaría los elementos HTML de tipo *h1* con los estilos de la **definición global**.

Existen más combinaciones pero con este ejemplo consideramos cumplidos los objetivos de este curso.

3.5 Propiedades en CSS

Ya hemos visto como se relacionan los elementos HTML con sus estilos y también hemos visto cómo y dónde definir los estilos. En los ejemplos mostrados hasta el momento, siempre hemos usado propiedades como COLOR, TEXT-ALIGN, BORDER, etc. aunque existen muchas más. En la siguiente tabla mostramos algunas de las principales (hay muchas más) propiedades de los distintos elementos con un valor a modo de ejemplo. Para una mayor especificación de las propiedades y valores que se pueden asignar mirar webs como http://www.htmldog.com/reference/csspropertiesl/

background-color: red;
background-image: url(imagen.jpg);
background-repeat: repeat-x;

border-top: 1px solid black;
border: 1em dotted #fc0;
border-style: dotted;
font: italic bold arial, Helvetica, sans-serif;
z-index: 2
display: block;
visibility: hidden;

No todos los navegadores representan igual todas las pantallas, e incluso no todos entienden todas las propiedades

No todas las propiedades tienen sentido en todos los elementos HTML. Las que no aplican a un elemento son ignoradas

4 Introducción a AJAX

Con lo que hemos visto hasta el momento, **la única forma de que una página Web se comunique con el servidor es enviando (submit) un formulario a un destino (una URL)**, definido este destino en la propiedad action del elemento HTML form.

Esta acción implica que **se realiza una petición a la que el servidor responde mandando, normalmente, la siguiente página web a mostrar**. Esta respuesta (response) es cogida por el navegador y pintada. Esto tiene **como consecuencia que la pantalla se recarga completamente**, no siendo muy placentero para el usuario y aumentando mucho el tráfico por la red.

Para solucionar estos problemas, entre otros, se creó **AJAX (Asynchronous JavaScript And XML)**. La idea de AJAX es que, mediante el uso de JavaScript, la página se puede comunicar con el servidor, cuya respuesta es tratada también con JavaScript, evitando así el tener que realizar la recarga de la página al completo.

Un **ejemplo típico** en el que se entienden muy bien los beneficios de AJAX es cuando en una página tienes **dos listas de valores en las que los valores de la segunda dependen del elegido en la primera**. Por ejemplo, la primera lista es de provincias y la segunda es de poblaciones de esa provincia. Cuando se cambia el valor elegido en la primera, la segunda debe recargarse con las poblaciones correspondientes a la provincia elegida. Si este problema lo resolviéramos sin AJAX habría que recargar la página completa de nuevo. Empleando AJAX, cuando cambia la provincia elegida se hace una petición en JavaScript y cuando el servidor responde con la lista de poblaciones, de nuevo con JavaScript se actualizan los valores de la lista de poblaciones sin necesidad de recargar la página consiguiendo así **mejora de rendimiento y usabilidad**.

En resumen, **mediante AJAX conseguimos aplicaciones Web más rápidas, sencillas y amigables para el usuario**.

Debido a que aún no hemos llegado a la parte del temario donde se explicará como hacer interactuar al cliente (navegador) con el servidor, vamos a posponer la explicación de AJAX a ese momento (Tema 4) para poder entender mejor sus beneficios.