Curso Web accesible con XHTML y CSS. Bloque I. XHTML

Cursos Thales CICA Web 2006. Curso WEBCSS

A. Gámez, L.M. Marín, F. Mesa & S. Fandiño

Índice del Bloque I. XHTML.

---

1. Introducción

Antes de entrar en materia

La gran mayoría de los usuarios, en la actualidad, acceden a Internet por medio de un navegador gráfico, y este suele ser Microsoft Internet Explorer en sus versiones más recientes. Esto no es verdad para todos los casos. Aún siendo una minoría, son muchas las personas que acceden utilizando otro navegador, otro sistema operativo, e incluso, navegadores no gráficos tales como un lector de pantalla o un dispositivo braille. Nuestro visitante ni siquiera tiene que ser una persona, podría ser el robot de algún buscador.

A pesar de esto, la gran mayoría de los 'diseñadores' de páginas web se concentran en que sus sitios se vean bien en la versión más reciente del navegador más popular. Dejando a un lado a los demás visitantes que han tenido la suerte o el interés en visitarlos.

Existen páginas realmente muy atractivas y sofisticadas desde un punto de vista estético. Pero la medida en que esa 'sofisticación' nos impide acceder a su información por medios diferentes a los tradicionales es la medida de su inaccesibilidad.

Cuando creamos un documento en una antigua máquina de escribir, ese pequeño renglón en la parte superior de nuestra página separado de todos los demás, sabemos que es un título. Es una cualidad de nuestro cerebro, capaz de identificar los patrones a su alrededor. Pero en el papel solo hay símbolos, que son difíciles de distinguir unos de otros. Cuando comenzamos a crear nuestros documentos digitales arrastramos esa tradición de dejarle sólo a nuestro cerebro el discernir qué es un título y qué no lo es. Pero en los medios informáticos tenemos un gran poder que la hoja de papel no tiene: la información puede hablar de la información. La información puede decir algo de sí misma. Un título no tiene porque ser solo una línea separada de todas las demás. Puede decirle al mundo (literalmente) que ella es un título, una lista, un párrafo, una cita, etc.

Un buen documento no solo contiene información, también incluye una estructura que dice algo sobre ella misma. Si esa estructura existe, su contenido se vuelve *accesible* y comprensible más allá del aprendizaje visual que nuestro cerebro tiene de las líneas en la parte superior de la página o las letras en *cursivas*. Se puede organizar y manipular en base a esa estructura de maneras imposibles de conseguir sin ella o en documentos en papel.

En la realidad muchos sitios web dejan de lado un orden lógico de la información anteponiendolo a una simple apariencia. Pero esto no tiene porqué ser así. Utilizando las herramientas adecuadas podemos satisfacer ambas necesidades. El no hacerlo es más una cuestión de voluntad que de dificultad.

Si no otorgamos a nuestros documentos de dichas estructuras, y le dejamos solo a nuestro cerebro el discernir qué es cada cosa por medio de la visión (un don que no todos tienen), estamos perdiendo un enorme potencial que la tecnología de la información nos ofrece y estaremos infrautilizándola. Solo estaremos haciendo lo mismo que se hacía antes de la aparición de los ordenadores personales, amontonando páginas y páginas de letras y símbolos extraños en grandes archivos digitales.

La Web, siendo un recurso de escala planetaria, debería ser accesible para todos. O al

menos, para todos los que puedan estar interesados en la información que se ofrece. El HTML es el lenguaje que nos facilitará otorgar la estructura que permite dicha accesibilidad con ayuda de los programas y dispositivos adecuados. Por supuesto que la apariencia es importante. Para influir sobre ella utilizaremos las hojas de estilo CSS.

Esperamos que este Curso pueda reflejar la filosofía del HTML.

---

2. ¿Cómo funciona la web?

Tal vez parezca una exageración, pero nosotros, quienes podríamos ser viejos usuarios de los ordenadores y de Internet, es posible que no tengamos del todo claros algunos conceptos importantes, aunque pensemos que así sea, sobre la WWW. Estas nociones son relevantes y dignas de ser tomadas en cuenta a la hora de construir nuestras propias páginas. Son 'cultura computacional' y nos dan plena conciencia de lo que en realidad estamos haciendo.

¿Qué es Internet?

Internet nos es otra cosa que una inmensa red de ordenadores que se comunican entre sí utilizando un mismo lenguaje, lo que permite que se entiendan unos a otros.

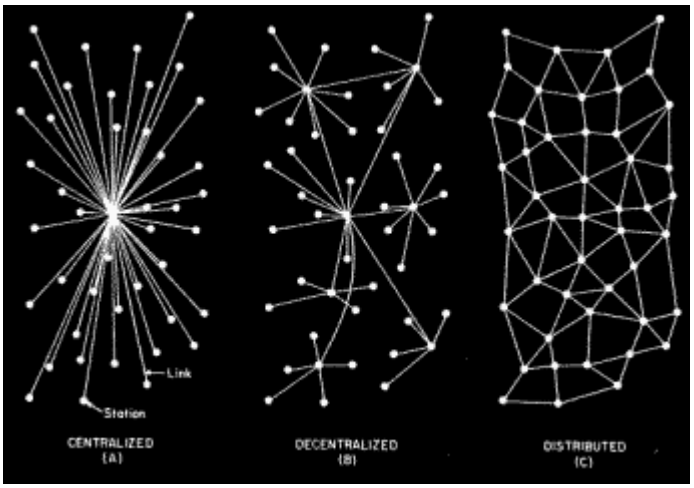


Imagen que muestra una red centralizada,
una descentralizada y una distribuida

No hay un 'Centro de Internet'. No hay un 'Ordenador Central', ni algo que tome el control o lo gobierne. Es una red distribuida. Cada uno de sus nodos es capaz de crear, enviar, y recibir información como cualquier otro. La razón se puede encontrar en sus orígenes, que tienen como base a una red concebida para ser funcional incluso después de un ataque nuclear. No hay un 'Centro de Control', ni máquinas más importantes que otras. No hay máquinas conectadas a ninguna máquina central. Todas están conectadas a sus vecinas, que a su vez están conectadas a sus propias vecinas. De tal manera que, siguiendo una ruta adecuada, puede llegar información de una máquina de la red a cualquier otra. Y no existe solo una forma de hacerlo.

Aunque prácticamente todos sus usuarios estamos acostumbrados a pensar en

Internet como un inmenso conjunto de páginas web, esto no es así. Internet no es sólo páginas web, tiene multitud de servicios. No son de nuestro interés aquí, pero es bueno tener conocimiento de ello. La parte de Internet que nos interesa es, por supuesto, la WWW.

¿Qué es la WWW?

Es el acrónimo de World Wide Web, que más o menos significa "Telaraña mundial". A partir de aquí hay algo que debemos recordar: *la Web son documentos enlazados unos con otros*. Esa es la esencia de la Web. Por supuesto tiene que haber algún mecanismo que haga eso posible. Pensemos en todo lo que tiene que suceder cada vez que hacemos algo tan fácil como un click a un vínculo con el ratón. Cuando hacemos click, el ordenador recibe la orden de buscar el documento que se le ha pedido. Así que tiene que ir hasta donde esté el documento en cuestión, recogerlo, y mostrarlo en la pantalla.

Si nos ponemos en el lugar del ordenador, lo primero que necesitamos es saber qué documento tenemos que ir a buscar. Necesitamos saber dónde tenemos que ir a buscarlo, y una vez localizado, tenemos que traer el documento de algún modo a través de las redes de comunicaciones, y por último, tenemos que ser capaces de representar el documento en la pantalla. Desde el punto de vista técnico hay muchas operaciones, que en realidad no deben preocuparnos, pero las mencionamos porque desde el punto de vista del diseñador esto es importante. De aquí salen algunos conceptos que son fundamentales.

En primer lugar, cada documento debe tener un localizador o un identificador, algo que diga cómo se llama el documento y dónde se encuentra, es decir, en qué ordenador de todos los conectados a Internet está almacenado. En segundo lugar, debe existir un lenguaje en que los ordenadores se comuniquen para pedirse y entregarse documentos unos a otros. Y en tercer lugar, debe existir una forma de codificar los documentos para que una vez que el ordenador, sea el que sea, lo haya obtenido, sea capaz de representarlo en la pantalla o en cualquier otro medio.

Vayamos por partes. Empecemos por el localizador, es el nombre único que recibe cada documento en la red. Es muy sencillo: es lo que se pone en la barra de direcciones del navegador. Por ejemplo: http://www.google.com/

De momento lo importante es que si queremos crear un documento que esté enlazado con otro, tenemos que hacer referencia a él con su nombre: http://www.google.com/

Lo siguiente es la comunicación entre ordenadores. No profundizaremos aquí, pero es relevante darnos cuenta que ya la hemos visto actuar. ¿Alguna vez te ha salido en la pantalla eso de "Error 404: Not found" o algo por el estilo?

¡Pues ahí está!. Los ordenadores tienen una especie de protocolo sencillo para comunicarse. Algo así: Me han dicho que tienes este documento. Dámelo, anda. Y la otra, si lo tiene, se lo da. Y si no lo tiene, le dice No lo encuentro.. Claro, los ordenadores hablan con números, y en vez de decir No lo encuentro dicen 404. Así que tu ordenador vuelve derrotado sin nada que entregar y un '404' en sus manos...

Este protocolo de comunicación consiste básicamente en unas cuantas reglas para que todo funcione bien, y unos cuantos códigos como "404", "500", "200", etc. Por cierto, el protocolo se llama HTTP, que significa **HyperText Transfer Protocol** (Protocolo para la

Transferencia de Hipertexto). Lo de "Protocolo para la transferencia" se entiende. ¿Y la parte de Hipertexto exactamente qué quiere decir? No se, pero mi formación me dice que Hiper es un prefijo que usan los matemáticos para indicar que algo tiene dimensiones extras. Como en hiperespacio o hipercubo. El texto normal tiene una sola dimensión, así que hipertexto será un texto con más de una dimensión. Entonces hipertexto es como el texto normal, pero mejor: puedes salirte del hilo, porque no estás restringido a una sola dimensión. ¡Puedes ir dando saltos!

La tercera parte es el mecanismo para codificar los documentos de modo que los programas que reciben el documento puedan descodificarlo y representarlo en la pantalla como lo que era: un documento de hipertexto. Si los documentos fueran de texto normal, bastaría con enviar un fichero con las palabras del texto, un documento de texto normal y corriente. Pero no es así, es hipertexto. Eso quiere decir que además de palabras y frases, hay vínculos que se refieren a otros documentos. Entonces necesitamos un lenguaje para formatear los documentos que tenga en cuenta que hay palabras y también que hay hipervínculos.

El HTML es precisamente ese lenguaje. Es el lenguaje que se creó para compartir documentos en la Web. En aquellos entonces los recursos eran limitados, así que tanto el protocolo HTTP como el lenguaje HTML tenían que ser muy sencillos: por eso son tan sencillos. HTML significa **HyperText Mark-up Language** (Lenguaje para el Formato de Documentos de Hipertexto) ¿Mark-up significa "formato de documentos? Básicamente es eso. Es un término de imprenta. Cuando un escritor escribía un libro, a mano o con una máquina de escribir, y se lo entregaba a su editor, el editor tenía que marcar sobre el texto instrucciones para que los de la imprenta imprimieran todo correctamente: decía dónde estaban los títulos, las secciones, marcaba los párrafos, etc. Todo eso lo anotaba con unas marcas más o menos estándares que los de la imprenta entendían. Al conjunto de todas esas marcas, en inglés se le llama "markup".

Eso es justamente lo mismo que vamos a hacer cuando escribamos en HTML. Haremos el rol de escritor, pues redactaremos nuestros documentos. Y el de editor, pues vamos a decir dónde acaba y dónde empieza cada párrafo, cuáles son los títulos, dónde acaba y dónde empieza una lista y cada elemento de la lista, etc. El navegador será como la imprenta, que va a reconocer todas esas marcas y le va a dar a nuestros documentos la apariencia deseada: los títulos más grandes, los párrafos separados, las listas con marcadores de lista, etc.

2. Diferencia entre HTML y XHTML

Originalmente solo existía el HTML. De éste han surgido diferentes versiones, y todas con una más o menos completa compatibilidad hacia atrás con las versiones anteriores. La última especificación HTML que existe y existirá es el HTML 4.01. Después de ella se desarrolló una nueva implementación llamada XHTML 1.0. Cuando hablamos de XHTML 1.0 estamos hablando de HTML 4.x pues son lenguajes idénticos, salvo algunas diferencias que serán aclaradas en su momento. Éste documento hace un uso indistinto de los términos HTML y XHTML. Y cuando lo hace se refiere a las versiones mencionadas arriba. Las versiones HTML 4.01 Estricta y XHTML 1.0 Estricta, que son las utilizadas aquí, prohíben el uso de entidades y atributos que sólo tengan un fin puramente de presentación y de

apariencia. Para el control de la apariencia se recurre al uso de tecnologías alternativas como son las Hojas de Estilo en Cascada CSS. De ahí que este Curso contenga tanto XHTML y CSS. Aclarado esto podemos comenzar.

---

3. El documento HTML (XHTML)

I. Las etiquetas

Los documentos XHTML no son más que textos etiquetados. Es con las etiquetas como indicamos al navegador o browser qué es cada una de las partes que lo componen.

Los tags o etiquetas pueden tomar las siguientes formas:

Etiqueta de apertura y cierre

<etiqueta1> ... </etiqueta1>

Todo lo que se encuentre entre las etiquetas se verá afectado por ellas.

Etiqueta única (solo válida en HTML):

<etiqueta2>

Esta etiqueta indica una acción que sólo se aplica en el lugar donde está ubicada. No pueden existir elementos dentro del elemento que representa.

Etiqueta única que cierra:

<etiqueta3 />

Es exactamente lo mismo que la etiqueta anterior, con una salvedad: en el lenguaje XHTML todas las etiquetas deben cerrarse de forma obligatoria a diferencia de lo que sucede en HTML. La barra hacia la derecha que se encuentra en la parte final de la etiqueta indica el cierre de la misma.

Aunque para el lenguaje HTML es indistinto el uso de mayúsculas o minúsculas, no sucede lo mismo para el XHTML donde es obligatorio poner las etiquetas en minúscula. Es ideal manejarlas así desde el principio, de esta manera nos evitaremos problemas a la hora de movernos entre diferentes implementaciones.

Cuando es definida una etiqueta, también es posible darle valor a sus atributos asociados. La manera de hacer esto se ilustra con los siguientes ejemplos:

<table class="miclase" summary="Este es el sumario">

<a href="http://www.google.com/">

En el primer ejemplo, la etiqueta <table> contiene a los atributos class y summary. Donde class toma el valor de miclase y summary toma el valor de Este es el sumario. En el segundo caso, la etiqueta <a> tiene un atributo llamado href que toma el valor de http://www.google.com/. No debemos preocuparnos ahora por el significado de estos atributos o las etiquetas. El propósito aquí es ilustrar la sintaxis básica que rige la declaración de etiquetas y sus atributos.

En XHTML al igual que los nombres de etiqueta, los atributos y sus valores deben ir en minúscula. Los valores a su vez deben ir entre comillas dobles. Siempre deben tener un valor y por tanto no se pueden minimizar a la antigua usanza.

Las etiquetas no pueden solaparse. Es obligatorio que un elemento se cierre dentro del elemento donde fue abierto. No se puede hacer algo como:

```
<strong><em>Texto con fuerza y énfasis</strong></em>
```

La forma correcta es:

```
<strong><em>Texto con fuerza y énfasis</em></strong>
```

Algunos atributos son obligatorios en su declaración, otros no lo son. Tampoco es verdad que todas las etiquetas poseen los mismos atributos, éstos dependerán de la etiqueta misma sobre la cual actúan.

Atributos comunes

Existen atributos que son comunes a todas las etiquetas. Para el HTML las únicas excepciones se encuentran en los elementos: <head>, <html>, <meta>, <param>, <script>, <style>, y <title>. Y estos son:

- accesskey: Permite la asignación de un carácter como tecla de método abreviado para activar el elemento. El valor debe ser un carácter.
- class: La clase del elemento. Permite distinguirlo de otros elementos iguales. Usualmente sirve para aplicar estilos CSS particulares. El valor es una cadena de caracteres.
- dir: Especifica la dirección del texto. Valores posibles: ltr, rtl.
- id: Especifica un identificador único para el elemento. Solo puede existir un mismo id por página. Debe ser una cadena de caracteres.
- lang: Indica el lenguaje del contenido de la etiqueta. El valor debe ser un código de lenguaje. es es el código para el español.
- style: Permite especificar reglas de estilo CSS en línea. No es recomendado hacerlo de esa manera. Los valores deben ser reglas de estilo.
- title: Permite especificar un título a la etiqueta. Tiene múltiples usos. Usualmente su contenido se muestra al dejar el cursor del ratón sobre el elemento en cuestión. Útil para abreviaturas o acrónimos. El valor debe ser una cadena de caracteres.
- tabindex: Un número que especifica el orden de tabulado del elemento.

Comentarios

Por último, si deseamos agregar algún comentario en nuestro documento HTML debemos hacer algo como esto:

```
<!-- Esto es un comentario -->
```

Como se muestra arriba, los comentarios se inician con <!-- y terminan con -->. Todo lo

que se encuentre dentro de esas marcas será ignorado por el navegador. Los comentarios pueden tener múltiples líneas, y no finalizan hasta que sea especificado, ignorando esos saltos de línea.

---

3. El documento HTML (XHTML)

II. La estructura del documento HTML (XHTML)

La estructura que todo documento HTML debe poseer es la siguiente:

```
<html>

 <head>

  <title>

  </title>

 </head>

 <body>

 </body>

</html>
```

Veremos cada una de sus partes:

<html> ... </html>

Todo el documento HTML se encontrará dentro de estas etiquetas.

<head> ... </head>

Esta es la cabecera del documento. Aquí se incluirá información importante sobre la página.

<title> ... </title>

El título de nuestra página.

<body> ... </body>

Este es el cuerpo de nuestro documento, aquí se colocará el contenido de nuestra página.

Mi primera página

Con lo que sabemos... y un poquito más, estamos en condiciones de escribir nuestra primera página:

```
<html>
```

```
<head>
 <title>
  Este es el título de mi página... (la primera de ellas)
 </title>
</head>
<body>
 <h1>
  Mi primera página
 </h1>
 <p>
  Este es el primer párrafo de mi primera página.
 </p>
 <p>
  Por ahora no tiene mucho, pero con el tiempo tendrá muchas cosas mas...
  Por cierto, este es el segundo párrafo.
 </p>
</body>
</html>
```

El resultado lo puedes ver a través del siguiente enlace: Ejemplo primera página.

Explicación:

<html>

Con ésta sentencia le indicamos a la computadora que el documento es un archivo en lenguaje HTML.

<head>

<title>

Este es el titulo de mi pagina... (la primera de ellas)

</title>

</head>

Esta es la cabecera de nuestro documento. Aquí se incluye información que habla acerca de la página y su contenido. En este caso, hemos incluido el título de nuestro documento dentro de las etiquetas <title> y </title>. Comúnmente ese título aparece en la barra de títulos del navegador.

<body> ... </body>

Con las etiquetas <body> y </body> indicamos el inicio y fin del cuerpo de nuestro documento. Todo lo que se encuentre dentro de estas etiquetas será nuestro documento en sí. Todo su contenido y la información que se desplegará en pantalla se encontrarán ahí.

<h1>

Mi primera página

</h1>

El elemento <h1> indica una cabecera de primer nivel. Es decir, un título de primer nivel. De la misma forma <h2> indicará uno de segundo nivel y así... hasta el sexto nivel. Debe hacerse notar que esta etiqueta cumple una función importante. No la usamos para que los títulos aparezcan más grandes que el resto del texto, lo hacemos para que este tenga una estructura lógica. Para que los títulos sean títulos y el texto sea texto. De esta manera, no sólo organizamos mejor nuestra información, sino que la hacemos más accesible a los demás. ¿Y su apariencia? Aunque por defecto aparecen más grandes que el resto del texto, nosotros podremos controlar su presentación utilizando hojas de estilo en cascada (CSS). Ese tema será abordado más adelante, a lo largo de los demás bloques del Curso.

Este es el primer párrafo de mi primera página.

Por ahora no tiene mucho, pero con el tiempo tendrá muchas cosas mas.

Las etiquetas y las utilizamos para indicar el inicio y fin de un párrafo. Estos párrafos suelen ser representados como secciones de texto separadas entre sí con un salto de línea de doble espacio. Podemos observar que no importa la forma en que hayamos escrito el texto que se encuentra dentro de las etiquetas. No importa si existen saltos de línea en nuestro código fuente. Estos saltos de línea no serán representados. El navegador sólo hará caso de las etiquetas para realizar esas tareas.

</html>

Marca el fin de nuestro documento HTML.

---

3. El documento HTML (XHTML)

III. El DOCTYPE y las versiones de HTML

Existe más de una versión de HTML. Aunque esencialmente son lo mismo, cambian un poco en su comportamiento y sentencias válidas. A partir de la versión 4.0 Estricta todas las etiquetas que tenían como función alterar la apariencia son desaprobadas y no forman parte del estándar. Ese es el HTML que vamos a estudiar en este Curso. El HTML sólo nos servirá para definir la estructura del documento y las hojas de estilo CSS para definir la apariencia del mismo.

Aunque es posible usar directamente la estructura especificada arriba a la hora de crear nuestras páginas, en la práctica es conveniente decirle al navegador qué versión de HTML pensamos utilizar en nuestro documento. En la gran mayoría de los navegadores esto alterará un poco la manera de interpretar y dibujar la página. Si queremos asegurar un comportamiento más o menos similar en todos los navegadores, será conveniente hacerlo. Esto se logra con el DOCTYPE. Éste se coloca al inicio del documento, antes de la etiqueta <html>. El que se corresponde con la versión que se utilizará aquí, el que estamos utilizando es el siguiente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Que se refiere a la versión XHTML 1.0 Estricta. La primera sección indica el número de versión a utilizar. La segunda parte es una referencia a la "definición de tipo de documento" (Document Type Definition).

Además, a nuestra etiqueta <html> la definiremos así:

```
<html xmlns="http://www.w3c.org/1999/xhtml" xml:lang="es">
```

El valor es del atributo xml:lang puede variar e indica el idioma del documento. En este caso se refiere al español.

Si algún procedimiento en particular de los que vamos a utilizar en este Curso no cumple los estándares del W3C (http://www.w3.org/), se especificará de antemano qué versión sí lo hace.

3. El documento HTML (XHTML)

IV. Los códigos de caracteres y los caracteres especiales

En los comienzos de la informática se creó un código llamado ASCII (American Standard Code for Information Interchange) para representar los caracteres. Este código asignaba a cada carácter un número (de 7 bits). Debido al tamaño de ese número (7 bits) ese código constaba únicamente de 128 caracteres que incluían las letras, números y algunos caracteres habituales. Estos caracteres los entiende cualquier ordenador y por tanto podemos usarlos con toda libertad en un documento HTML. Con la evolución y expansión de los ordenadores pronto surgió la necesidad de añadir nuevos caracteres, como por ejemplo aquellos específicos de cada idioma. Por esta razón se crearon extensiones del código ASCII que añadían un bit al número asociado a cada carácter con lo que el número de caracteres representable aumentaba a 256. El problema es que no hubo un acuerdo en este ASCII extendido y se crearon varias versiones entre las que había ligeras diferencias. Los documentos HTML pueden tener todo tipo de caracteres pertenecientes al ISO-Latin-1 (ISO-8859-1) que es una de estas extensiones. Aún así los caracteres que no pertenezcan al código ASCII de 7 bits deben introducirse con la secuencia de escape correspondiente para evitar problemas con ordenadores que no soporten la codficación ISO-Latin-1 sino otra extensión del ASCII. El estándar HTML 4.0 ha llevado más lejos la extensión del código ASCII soportando el conjunto de caracteres UNICODE. Este conjunto incluye los caracteres del ASCII (a los que asigna el mismo número y por tanto es compatible), pero añade todos los caracteres de todos los idiomas del mundo (incluidos los chinos y japoneses). Esto es importante porque UNICODE se está imponiendo como uno de los estándares del presente y del futuro.

Cuando nosotros creamos un documento HTML, es importante señalar que conjunto de caracteres estamos utilizando. Esto lo haremos con la siguiente línea de código:

```
<meta http-equiv="Content-Type" content="text/html; charset=us-ascii" />
```

Es la definición de un 'metadato', que no es sino información sobre nuestra página HTML. Se colocará entre las etiquetas <head> y </head> de nuestro documento. En este caso estamos señalando que usaremos el conjunto de caracteres ASCII estándar.

Repasando lo que hemos visto sobre el DOCTYPE y los códigos de caracteres, nuestros documentos HTML deberán tener la siguiente estructura:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es">
```

```
<head>
 <meta http-equiv="Content-Type" content="text/html; charset=us-ascii">
 <title>
 </title>
</head>
<body>
</body>
</html>
```

De esta forma declaramos el uso del estándar XHTML 1.0 Estricto y el uso del juego de caracteres ASCII.

¿Y qué pasa con los caracteres no-ASCII?

En caso de requerir de caracteres que no formen parte de la especificación ASCII (y en general, de cualquier conjunto de caracteres declarado en nuestro documento) es necesario usar secuencias de escape que representen dichos caracteres.

Si queremos utilizar cualquier tipo de caracteres, deberemos utilizar el formato Unicode que se consigue utilizando:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Hay muchas maneras de representar el mismo carácter con secuencias, pero las más comunes hacen uso de nemotécnicos y valores numéricos decimales.

Secuencias de escape numéricas

Las referencias numéricas de caracteres especifican la posición del código de un carácter en el conjunto de caracteres del documento. Las referencias numéricas de caracteres pueden tener dos formas:

La sintaxis "&#D;", donde D es un número decimal, se refiere al carácter de ISO-10646 con el número decimal D.

La sintaxis "&#xH;" o "&#XH;", donde H es un número hexadecimal, se refiere al carácter de ISO 10646 con el número hexadecimal H. Para los números hexadecimales de referencias de caracteres numéricas no se distingue entre mayúsculas y minúsculas.

A continuación se muestran algunos ejemplos de referencias numéricas de caracteres:

- å (en decimal) representa la letra "a" con un circulito encima

(usada, por ejemplo, en noruego).

- å (en hexadecimal) representa el mismo carácter.
- å (en hexadecimal) representa también el mismo carácter.
- И (en decimal) representa la letra mayúscula cirílica "I".
- 水 (en hexadecimal) representa el carácter chino para "agua".

Para consultar la tabla de todos los caracteres ISO-10646 - UNICODE con su respectivo valor numérico puedes visitar: http://www.unicode.org/charts/

Secuencias de escape nemotécnicas

Aprenderse un conjunto de valores numéricos para cada uno de los carácteres especiales que pretendamos usar es una tarea complicada. Para resolver ese problema existen ciertas secuencias de escape nemotécnicas para los más usuales (pero debe aclararse que no todos los caracteres especiales tienen un equivalente nemotécnico).

Entre los más usuales podemos encontrar:

Carácteres especiales

Letras acentuadas, diéresis, etc.

Á Á	À À	Â Â	Ä Ä	Ã Ã Å Å
á á	à à	â â	ä ä	ã ã å å
É É	È È	Ê Ê	Ë Ë	
é é	è è	ê ê	ë ë	
Í Í	Ì Ì	Î Î	Ï Ï	
í í	ì ì	î î	ï ï	
Ó Ó	Ò Ò	Ô Ô	Ö Ö	Õ Õ
ó ó	ò ò	ô ô	ö ö	õ õ
Ú Ú	Ù Ù	Û Û	Ü Ü	
ú ú	ù ù	û û	ü ü	
Ý Ý	ý ý	ÿ ÿ		
ñ ñ	Ñ Ñ	Ç Ç	ç ç	

Signos de puntuación

¡ ¡	¿ ¿	´ ´	· ·	¸ ¸
« «	» »	¨ ¨		

Otros alfabetos

Æ Æ	æ æ	ß ß	µ µ
Ð Ð	ð ð	Þ Þ	þ þ

Monedas

¢ ¢ £ £ ¤ ¤ ¥ ¥

Símbolos matemáticos

¹ ¹ ² ² ³ ³
 × × ÷ ÷
± ± ¼ ¼ ½ ½ ¾ ¾
Ø Ø ø ø
¬ ¬

Operadores empleados por el código HTML

< < > > & &   " "

Otros símbolos

º º ª ª © © ® ® ° °
¦ ¦ § § ¶ ¶ ¯ ¯

Si no es posible ver algunos carácteres probablemente se deba a que no son visibles (como un espacio), o bien, que el ordenador no soporta ese carácter con la configuración actual.

---

4. Elementos del XHTML

I. Elementos de texto

Una vez que tenemos nuestra estructura base estamos listos para llenar nuestra página de información. Como la gran mayoría de ésta se encuentra en forma de texto, los primeros elementos que veremos se relacionan estrechamente con él.

Los principales son:

Elemento	Etiqueta	Es de tipo:
Cabeceras	<h1> ... <h6>	bloque
Párrafo	<p>	bloque
Sálto de línea	 	-
Preformateado	<pre>	bloque
Cita	<q>	línea
Cita en bloque	<blockquote>	bloque
Énfasis		línea
Fuerza		línea
Código	<code>	línea
Acrónimo	<acronym>	línea

Abreviatura <abbr> línea

Lo mejor es verlos en acción. Haz click en el vínculo de abajo para verlo. No olvides revisar el código fuente para ver como está construido todo. Una manera de ver el código fuente es hacer click con el botón secundario en la página y seleccionar Ver código fuente, o algún comando equivalente en tu navegador. Puedes usar distintos navegadores para diferenciar el aspecto.

Haz click aquí para ver el ejemplo

Como podrás notar algunos elementos se representan de forma diferente por el simple hecho de ser una entidad u otra. Los párrafos se muestran de forma diferente que los títulos o el texto con énfasis. Otros, por el contrario, no muestran diferencia alguna en su representación, como las citas o las abreviaturas, a no ser que indiquemos explícitamente lo contrario. (Esto también depende del navegador utilizado)

Atributos

Los siguientes elementos poseen atributos exclusivos:

Cita <q>

- cite: Define el origen de la cita.

Cita en bloque <blockquote>

- cite: Define el origen de la cita.

¿La página del ejemplo parece demasido sobría y sin chiste? Tal vez. Eso se puede arreglar con las hojas de estilo, que nos permitirán controlar la apariencia y comportamiento de nuestros elementos. Si lo que deseas es controlar más a fondo su apariencia podrás hacerlo más adelante en el curso, en la sección de hojas de estilo.

---

4. Elementos del XHTML

II. Vínculos

Los vínculos o enlaces son la escencia de la Web, aquí aprenderemos algo relativo a los mismos.

El elemento que nos permite la creación de vínculos o anclas es <a>. Este elemento sirve para redirigirnos a otro recurso del Web, pero también se puede convertir en el ***destino*** de un vínculo externo.

El ancla <a> es por defecto un elemento de línea. Recordemos que los elementos de línea no alteran el flujo del texto que se encuentra a su alrededor, a diferencia de los elementos de bloque, que añaden un salto de línea antes y al final de ellos mismos.

Veamos los diferentes usos que tiene el elemento <a>:

Como vínculo a otro recurso

La sintáxis es la siguiente:

```
<a href="http://www.google.com/">Ir a Google</a>
```

El atributo href indica el destino del vínculo. Pueden utilizarse direcciones absolutas (como en el ejemplo), o bien direcciones relativas con respecto al directorio en que se encuentra el documento. Y en general, puede usarse cualquier URI válido. El vínculo no solo permite crear un enlace hacia otra página web. ¡Permite hacer un vínculo a cualquier recurso imaginable! Entre estos se encuentran las referencias a **http**, **ftp**, o **mailto**, etc. Es posible crear un enlace a cualquier tipo de archivo, no es indispensable que sean páginas web. Pero la manera en que el navegador se comportara ante esos archivos al activar el vínculo dependerá de su configuración. Usualmente aparece un cuadro de diálogo que pregunta al visitante qué desea hacer con dicho recurso. Se debe tener presente que no se puede asegurar un mismo comportamiento para todos los casos, ni para todos los navegadores.

Rutas absolutas y relativas

Una ruta absoluta para un recurso web es del tipo: http://www.dominio.com/carpeta/subcarpeta/documento.html. En el cual indicamos el nombre del ordenador y la ruta dentro de ese ordenador donde se encuentra el documento buscado.

Si nuestra página se encontrase en el directorio ***carpeta*** en la máquina del ejemplo de arriba, bastaría con utilizar una ruta relativa para acceder a ***documento.html***, que seria: subcarpeta/documento.html.

También es posible hacer referencia a directorios padre. Los dos puntos seguidos: .. hacen referencia al directorio de nivel superior. Si desde la misma página yo quisiera acceder al archivo ***documento2.html*** ubicado en el directorio raíz del dominio bastaría con escribir la ruta: ../documento2.html.

Estas reglas se aplican para cualquier elemento que haga uso de una dirección, no son exclusivas del elemento <a>.

¿Qué puede contener?

Cualquier elemento de línea puede estar contenido entre las etiquetas <a> y . ***Pero ninguno de bloque***.

Como destino de un vínculo externo

La sintáxis es la siguiente:

```
<a id="midestino">Quiero ésto como destino</a>
```

La posición en el documento que tenga un vínculo como el de arriba será el destino del ancla que lo mandó llamar. Dicha ancla deberá tener una sintaxis como la siguiente:

```
<a href="pagina.html#midestino">Ir al destino</a>
```

El ejemplo valdrá si el ancla destino se encuentra en el archivo ***pagina.html***. De hecho, ambos pueden estar en la misma página, permitiendonos dezplazarnos por ella de manera más eficiente.

Lo mejor es verlos en acción. Haz click en el vínculo de abajo para verlo. No olvides revisar el código fuente para ver como está construido todo. Una manera de ver el código

fuente es hacer click con el botón secundario en la página y seleccionar Ver código fuente, o algún comando equivalente de tu navegador.

Haz click aquí para ver el ejemplo 3

Atributos

Los atributos del elemento <a> son muchos. Aquí solo los enumeraré y comentarémos algunos de ellos más adelante:

Ancla <a>

- charset: Indica la codificación de caracteres del documento destino.
- coords: Permite especificar las coordenadas que definirán una región para un mapa de imagen.
- href: Un URI que especifica el destino del vínculo.
- hreflang: Indica la lengua base del documento destino. Por ejemplo: es
- rel: Indica la relación del documento destino respecto al documento actual. Valores posibles: alternate, designates, stylesheet, start, next, prev, contents, index, glossary, copyright, chapter, section, subsection, appendix, help, bookmark.
- rev: Indica la relación del documento actual respecto al documento destino. Valores posibles: alternate, designates, stylesheet, start, next, prev, contents, index, glossary, copyright, chapter, section, subsection, appendix, help, bookmark.
- shape: Especifica el tipo de región a emplear en un mapa de imágen. Valores posibles: rect, rectangle, circ, circle, poly, polygon.
- type: Establece el MIME del recurso destino.

Los mapas de imágenes los abordaremos más adelante.

---

4. Elementos del XHTML

III. Imágenes

Las imágenes son un recurso valiosísimo a la hora de presentar información de una forma clara, y lo es también como elemento ornamental.

El elemento que nos permite incluír una imagen es <img>. Por defecto las imágenes así incrustadas se comportan como elementos de línea.

Una etiqueta <img> típica aparecerá más o menos así:

```
<img alt="Un texto alternativo para la figura" src="imagen.gif" height="100" width="100" />
```

Atributos

Los atributos del elemento <img> son los siguientes:

Imágen <img>

- alt: Una descripción de la imagen (obligatorio).
- height: La altura de la imagen. Puede ser indicada en pixeles o porcentaje.
- ismap: Un URL que se refiere a un mapa de imagen del lado del servidor.
- longdesc: El URL de una página con una descripción larga de la imagen.
- src: Un URL con la ruta a la imagen.
- usemap: Un URL que indica un mapa de imagen del lado del cliente.
- width: El ancho de la imagen. Puede ser indicado en pixeles o porcentaje.

En principio se puede usar cualquier formato de imagen con la etiqueta <img>. Sin embargo, no todos los formatos son accesibles por todos los navegadores. Pero existen tres que podemos considerar universales en este momento y que todos los navegadores visuales soportan:

- JPEG
- GIF
- PNG

Con estos formatos aseguramos que nuestros visitantes podrán observar nuestras imágenes. Todos los formatos anteriores son de tipo bitmap.

Y ahora, la obligada página de ejemplo a continuación:

Haz click aquí para ver el ejemplo 4.

No olvídes revisar el código fuente de la página.

---

4. Elementos del XHTML

IV. Contenedores (capas)

La única función que poseen los contenedores es la de agrupar elementos. Ya sean de línea o bloque. Visualmente no otorgan ninguna característica particular a los elementos que contienen. Al menos, no por defecto. Su verdadera utilidad está en su uso combinado con las hojas de estilo, que nos permitirá aplicar reglas particulares a los grupos en cuestión.

Los contenedores son los siguentes:

Elemento	Etiqueta	Es de tipo:
Contenedor de bloque		bloque
Contenedor de línea		línea

No poseen atributos exclusivos.

Su utilidad como herramientas para la maquetación se abordará en los bloques y secciones de CSS de este Curso.

Usos recomendados

Las agrupaciones que nos otorgan estos elementos son geniales y muy útiles. Son los prototipos de elementos en bloque y elementos en línea, respectivamente. Con la suficiente habilidad y conocimiento, se podría crear una web entera utilizando sólo estos dos elementos y las CSS. Sin embargo, no es recomendable hacerlo así, ya que perderíamos el componente estructural que los otros elementos otorgan.

Cuando utilizamos los y los para aplicar estilos a un conjunto de elementos, usualmente asignaremos un valor al atributo class de cada uno de ellos. Es conveniente utilizar un nombre que esté relacionado con su contenido. Por ejemplo, si nuestro elemento contendrá el menú de navegación y algunos enlaces, tal vez sea conveniente llamarlo navegacion. Si colocamos el título de nuestra página, tal vez sea conveniente llamarlo cabecera. No es recomendable usar nombres que definan su apariencia visual (salvo muy contadas excepciones), como podría ser centrado, itálica, etc. Tal vez en ese momento dicho contenido se encuentre en efecto centrado. Pero si en el futuro decidimos cambiar el estilo y apariencia de nuestro sitio, es posible que el elemento que antes se encontraba centrado después se muestre a la izquierda, lo que nos conduciría a una incongruencia en la nomenclatura. O peor: elementos que tenían en común estar centrados perteneciendo a la clase centrado no compartan su tipo de alineación después. Lo que nos obligaría a editar el código HTML de todas las páginas involucradas. Además, ese tipo de prácticas nos hace proclives a cometer errores en la estructuración de nuestras páginas. Es mucho mejor denominar a las clases de acuerdo con su contenido y no con su apariencia visual.

---

4. Elementos del XHTML

V. Listas

Como su nombre lo indica, estos elementos nos serán muy útiles para listar cosas. Existen tres tipos de listas: Ordenadas, no ordenadas, y listas de definición. Dentro de los elementos que definen una lista se encuentran los elementos listados o *elementos de lista*. Veamos esto con los siguientes cuadros:

Elementos para definir listas

Elemento	**Etiqueta**	**Es de tipo:**
Lista ordenada		bloque
Lista no ordenada		bloque
Lista de definición	<dl>	bloque

Para definir su contenido se usan los siguientes elementos:

Para definir elementos de lista

Elemento	**Etiqueta**	**Es de tipo:**
Elemento de lista		bloque

Elemento a definir	<dt>	bloque
Definición	<dd>	bloque

No está de más decir que los elementos de lista <li> se pueden usar tanto en listas ordenadas como no ordenadas. Los elementos <dt> y <dd> sólo se pueden usar en listas de definición.

Ninguno de estos elementos posee atributos exclusivos.

Sintaxis básica

Para una lista ordenada:

```
<ol>

 <li>Primer elemento de la lista</li>

 <li>Segundo elemento de la lista</li>

 <li>Tercer elemento de la lista</li>

</ol>
```

Y se verá más o menos así:

1. Primer elemento de la lista
2. Segundo elemento de la lista
3. Tercer elemento de la lista

Para una lista no ordenada:

```
<ul>

 <li>Primer elemento de la lista</li>

 <li>Segundo elemento de la lista</li>

 <li>Tercer elemento de la lista</li>

</ul>
```

Y se verá más o menos así:

- Primer elemento de la lista
- Segundo elemento de la lista
- Tercer elemento de la lista

Es posible anidar listas ordenadas y no ordenadas entre sí. De manera que un elemento de lista puede contener a otra lista dentro de ella. Hay que tener mucho cuidado con el anidamiento, pues las listas no deben nunca solaparse. Sí podrá estar una lista dentro de otra, pero nunca solapadas.

Para una lista de definiciones:

```
<dl>

 <dt>Primer termino a definir</dt>

  <dd>Primera definición</dd>

 <dt>Segundo termino a definir</dt>

  <dd>Segunda definición</dd>

</dl>
```

Obteniendo algo como esto:

Primer término a definir
    Primera definicion
Segundo término a definir
    Segunda definición

Haz click aquí para ver el ejemplo de listas

---

4. Elementos del XHTML

VI. Tablas

Las tablas son un recurso muy valioso cuando tenemos la necesidad de presentar datos en forma tabular. Son un tema extenso y se podría crear un pequeño curso dedicado exclusivamente a ellas.

Una pequeña reseña

En los inicios del lenguaje HTML no había manera de controlar la apariencia visual de sus elementos, salvo utilizando el comportamiento por defecto que presentaban muchas de sus etiquetas. Los títulos se utilizaban para mostrar texto grande o en negritas, y no para su propósito original. Poco después aparecieron etiquetas que tenían un fin únicamente de presentación, mezclando así estructura con presentación. Muchos aún tienen la manía de

pensar que el HTML sirve para *diseñar*. Era (y sigue siendo) común recurrir a trucos *sucios* para conseguir efectos visuales y maquetar las páginas. Uno de los elementos más recurridos para ello lo encontramos en el uso de las tablas.

La tabla se ha llegado a utilizar para crear columnas o intentar colocar con precisión *pixelmétrica* elementos en las páginas. Una manera de conseguirlo ha sido anidando tablas unas dentro de otras, hasta obtener el resultado deseado. ¿Y qué ha pasado con la estructura? Nada, que se se dejado en el camino. Empezamos a tener elementos sepultados bajo líneas y líneas de código, generalmente celdas y definiciones de tablas, solo para colocar el contenido *en un lugar*, sin el más mínimo respeto por el orden lógico de la información.

El HTML parecía quedarse corto, pero solo se estaba utilizando en cosas para las cuales no fue diseñado. En esos tiempos era comprensible que las personas recurrieran a semejantes artimañas para hacer más vistoso su sitio. Ahora no. Las CSS han nacido para paliar esa carencia, y separar la estructura de la presentación.

Administrar un sitio maquetado con tablas es una pesadilla. Un vez construidas es muy difícil y laborioso modificar su aspecto visual. Hacer pequeños cambios implica la edición de muchas líneas de código. Las CSS permiten hacer todo más fácil y rápido. Recomendamos usarlas desde el principio y acostubrarse a ellas, sin importar lo que puedan hacer otros. Pensamos que el uso combinado de un documento HTML bien estructurado que haga uso de las hojas de estilo, son la combinación perfecta para conseguir una excelente página web.

Los elementos de tabla

Las tablas son entidades complejas que vale la pena conocer bien. Aquí veremos sus aspectos más comunes y utilizados. Serán suficientes para la mayoría de nuestras necesidades. Los elementos que debemos conocer son:

Elemento	Etiqueta	Es de tipo:
Tabla	<table>	bloque
Título de la tabla	<caption>	bloque
Fila	<tr>	bloque
Celda cabecera	<th>	bloque
Celda de datos	<td>	bloque

Existen otros elementos relacionados con las tablas que tienen la función de agrupar celdas o filas, otorgando de esta manera más información estructural a la misma. Sin embargo, para la gran mayoría de nuestros propósitos, los elementos mencionados arriba serán más que suficientes.

Elementos de agrupamiento:

Elemento	Etiqueta	Es de tipo:
Cabecera de tabla	<thead>	bloque

Grupo de columnas	<colgroup>	bloque
Columna	<col>	bloque
Grupo de filas	<tbody>	bloque

El dilema de la tablas

Las tablas, a diferencia de otros elementos, conservan todavía algunos de sus atributos con fines de presentación. Tiene sentido si nos ponemos a pensar que, en la mayoría de las ocasiones, los estilos aplicados a una tabla suelen usarse sólo una vez. Hemos decidido incluir en este primer bloque del curso todos los atributos que tienen fines para presentación, pero sólo vamos a profundizar en su uso para aquellos que sea imposible, muy difícil, o impráctica, su implementación utilizando CSS.

Sintaxis básica

La estructura básica de una tabla se presenta a continuación:

```
<table summary="Una tabla ejemplo con datos sin sentido">
 <caption>El nombre de la tabla</caption>
 <tr>
  <th>Celda cabecera 1</th>
  <th>Celda cabecera 2</th>
  <th>Celda cabecera 3</th>
 </tr>
 <tr>
  <td>Celda de datos 1</td>
  <td>Celda de datos 2</td>
  <td>Celda de datos 3</td>
 </tr>
 <tr>
  <td>Celda de datos 4</td>
  <td>Celda de datos 5</td>
  <td>Celda de datos 6</td>
 </tr>
</table>
```

El código superior nos crearía una tabla como la mostrada a continuación:

El nombre de la tabla

Celda cabecera 1	**Celda cabecera 2**	**Celda cabecera 3**
Celda de datos 1	Celda de datos 2	Celda de datos 3

Celda de datos 4 Celda de datos 5 Celda de datos 6

La tabla de arriba está afectada por la hoja de estilos que se aplica a esta página. Lo hemos dejado así para que exista una diferencia visual entre las celdas cabecera y las celdas de datos. Toda la aperiencia puede controlarse con las hojas de estilo CSS.

Explicación

La etiqueta <table> define la tabla. Incluye un atributo llamado summary. Dicho atributo es obligatorio. Su objetivo es contener un resumen del contenido de la tabla. No es posible apreciarlo con un navegador visual, pero su función radica en brindar información sobre la tabla para la navegación no visual. Este es otro aspecto más relacionado con la accesibilidad de nuestras páginas web.

La etiqueta <caption> contiene el título de la tabla. No puede existir más de una por tabla. No es obligatoria.

Las etiquetas <tr> definen las filas de la tabla. Es obligatorio que la tabla tenga por lo menos uno de estos elementos.

Las etiquetas <td> y <th> definen celdas. Debe existir por lo menos una para cada fila de la tabla. No es obligatoria la existencia del mismo número de celdas por fila.

Agrupando

Los elementos <col> y <colgroup> se colocan delante de la definición de las filas. Usualmente los elementos <col> van dentro de los elementos <colgroup> Su número y orden de aparición representa el orden y número de columnas que afectan. Por el contrario, los elementos <tbody> y <thead> siempre contienen dentro al conjunto de filas que afectan.

Un ejemplo:

```
<table summary="Una tabla ejemplo con datos sin sentido">
 <caption>El nombre de la tabla</caption>
 <colgroup span="2">
  <col align="right"></col>
  <col valign="top"></col>
 </colgroup>
<thead>
 <tr>
  <th>Celda <br />cabecera 1</th>
  <th>Celda cabecera 2</th>
  <th>Celda cabecera 3</th>
 </tr>
</thead>
 <tbody>
 <tr>
  <td>Celda de <br />datos 1</td>
```

```
 <td>Celda de datos 2</td>
 <td>Celda de datos 3</td>
</tr>
</tbody>
<tr>
 <td>Celda de <br />datos 4</td>
 <td>Celda de datos 5</td>
 <td>Celda de datos 6</td>
</tr>
</table>
```

Lo que nos da como resultado esto:

El nombre de la tabla

Celda cabecera 1	**Celda cabecera 2**	**Celda cabecera 3**
Celda de datos 1	Celda de datos 2	Celda de datos 3
Celda de datos 4	Celda de datos 5	Celda de datos 6

El elemento <colgroup> nos han permitido agrupar las dos primeras columnas. Los elementos <col> aplicar diferentes alineaciones verticales y horizontales a cada una de ellas. Los <tbody> y <thead> agrupan filas, pero no realizan ninguna acción particular sobre ellas.

Atributos

Se enumeran a continuación todos los atributos exclusivos aceptados por el estándar. Sin embargo, los atributos que sólo tienen fines para presentación no vamos a abordarlos aquí:

Tabla <tabla>

- border: Especifica en pixeles el grosor del borde de la tabla. Sin embargo, pueden usarse hojas de estilo para ello.
- cellpadding: Especifica en pixeles o porcentaje el espaciado que debe existir entre las celdas y su contenido. Las hojas de estilo ofrecen un control más poderoso de ese aspecto.
- cellspacing: Especifica en pixeles o porcentaje el espacio que debe existir entre celdas. Las hojas de estilo ofrecen un control más poderoso de ese aspecto.
- frame: Especifica cómo debe presentarse el borde de la tabla. Valores posibles: void, above, below, hsides, lhs, rhs, vsides, bos, border. Las hojas de

estilo permiten controlar ese aspecto también.

- rules: Indica que líneas deben ser dibujadas. Valores posibles: none, groups, rows, cols, all. Puede ser sustituida por hojas de estilo.
- summary: Contiene un resumen del contenido de la tabla. Es obligatorio.
- width: Indica en pixeles o porcentaje el ancho de la tabla. También se pueden usar las hojas de estilo para ese propósito.

Fila <tr>

- align: Indica la alineación horizontal del contenido de las celdas. Valores posibles: right, left, center, justify, char. Las hojas de estilo pueden utilizarse para reemplazar la mayoría de sus posibles comportamientos.
- char: El carácter que será utilizado como pivote de alineación. El atributo align debe ser igual a char.
- charoff: Indica en pixeles o porcentaje el offset de la alineación basada en carácter.
- valign: Especifica la alineación vertical de los datos en las celdas. Valores posibles: top, middle, bottom, baseline. Se pueden utilizar hojas de estilo para definir ese comportamiento

Celdas <th> y <td>

- abbr: Una versión abreviada del contenido de la celda.
- align: Indica la alineación horizontal del contenido de las celdas. Valores posibles: right, left, center, justify, char. Las hojas de estilo pueden ser utilizadas para reemplazar la mayoría de sus posibles comportamientos.
- axis: Define un nombre a la celda.
- char: El carácter que será utilizado como pivote de alineación. El atributo align debe ser igual a char.
- charoff: Indica en pixeles o porcentaje el offset de la alineación basada en carácter.
- colspan: Indica el número de columnas que esa celda debe expanderse.
- headers: Una lista de elementos separados con un espacio que incluye las Id's de celdas que provean información de cabecera para la celda activa. Útil para los navegadores no visuales.
- rowspan: Indica el número de filas que esa celda debe expanderse.
- scope: Señala si la celda provee información de cabecera para la fila, columna, grupo de filas, o grupo de columnas que la contiene. Valores posibles: col, colgroup, row, rowgroup.
- valign: Especifica la alineación vertical de los datos en las celdas. Valores posibles: top, middle, bottom, baseline. Se pueden utilizar hojas de estilo

para definir ese comportamiento

Agrupación de filas <thead> y <tbody>

- align: Indica la alineación horizontal del contenido de las celdas. Valores posibles: right, left, center, justify, char. Las hojas de estilo pueden ser utilizadas para reemplazar la mayoría de sus posibles comportamientos.
- char: Especifica el carácter que se utilizará como pivote en la alineación basada en carácter. El atributo align debe ser igual a char.
- charoff: Indica en pixeles o porcentaje el offset de la alineación basada en carácter.
- valign: Especifica la alineación vertical de los datos en las celdas. Valores posibles: top, middle, bottom, baseline. Se pueden utilizar hojas de estilo para definir ese comportamiento

El obligado ejemplo de ésta sección:

Haz click aquí para ver el ejemplo

También se pueden incluir formularios, incrustar objetos, etc. pero esto será objeto de estudio en otros bloques de contenidos de este curso.

---

4. Elementos del XHTML

VII. Elementos de formulario

Un formulario HTML es una sección del documento en que los usuarios pueden introducir datos, o bien, modificar controles y parámetros, que posteriormente serán enviados a un agente para que los procese.

Los formularios permiten la existencia de cierta 'interactividad' entre el visitante y la página web. Aunque ésta no es muy grande, puede resultar útil en muchas ocasiones.

Una nueva reseña

Al igual que la tabla, los formularios tienen la tendencia ser usados mal (o mejor dicho: los creadores de páginas web tienden a usarlos mal). Los formularios, si bien forman parte del HTML, la manera en que sus datos son gestionados no. Lo más habitual es utilizar lenguajes del lado del servidor para realizar esta tarea. O bien, recursos del lado del cliente si los parámetros no necesitan ser enviados a ningún lugar.

Ya que esa gestión no forma parte del HTML, escapa de los propósitos de este curso y por tanto no la vamos a abordar aquí.

El elemento *form*

La definición en el documento de un formulario depende del elemento <form>. Un formulario puede contener texto y código además de *controles de formulario*. Este elemento es de bloque.

Atributos del elemento *form*

Formulario <form>

- action: Especifica un URI que hace referencia a un agente procesador de formularios.
- accept: Este atributo especifica una lista de tipos de contenido separados por comas que un servidor procesador de formularios manejará correctamente.
- accept-charset: Este atributo especifica la lista de codificaciones de caracteres para los datos introducidos que son aceptadas por el servidor que procesa este formulario. El valor es una lista de valores de codificaciones de caracteres separadas por espacios y/o comas. El valor por defecto es unknown.
- enctype: Este atributo especifica el tipo de contenido usado para enviar el formulario al servidor (cuando el valor del atributo method sea post). El valor por defecto de este atributo es application/x-www-form-urlencoded. El valor multipart/form-data debería usarse en combinación con el elemento <input type="file">.
- method: Indica el método que el HTTP utilizará para enviar los datos del formulario. Valores posibles: get y post. Utilizando get, el conjunto de datos del formulario se agrega al URI especificado por el atributo action (con un signo de interrogación ("?") como separador) y este nuevo URI se envía al agente procesador. Utilizando post, el conjunto de datos del formulario se incluye en el cuerpo del formulario y se envía al agente procesador.

Un sencillo ejemplo de lo expuesto anteriormente. Muestra un formulario que va a ser procesado por el programa "agente" cuando sea enviado. El formulario será enviado al programa usando el método HTTP post.

```
<form action="http://www.unsitio.com/agente" method="post">
 <!-- Contenidos del formulario -->
</form>
```

El usuario de un formulario interactúa con él por medio de unos elementos llamados *controles*.

El elemento *input*

Para controles que permitan la introducción de datos por parte del usuario lo más usual es recurrir al elemento input. Éste elemento nos permitirá introducir en nuestro documento diversos tipos de controles de formulario. Éste elemento es vacío, por lo tanto solo basta una etiqueta que cierra para definirlo. Su comportamiento por defecto es como elemento en línea. Primero veamos sus atributos:

Atributos

Entrada <input>

- accept: Este atributo especifica una lista de tipos de contenido separados por comas que un servidor procesador de formularios manejará correctamente. Solo deberá usarse en caso de que el atributo type sea igual a file.
- alt: Muestra un texto alternativo para las imágenes. Solo aplica cuando el atributo type es igual a image.
- checked: Indica si el control esta marcado o no. Valor posible: checked. Solo tiene sentido cuando el atributo type es igual a checkbox o radio.
- disabled: Especifica si el control está deshabilitado o no. Valor posible: disabled. No puede ser usado cuando el atributo type es igual a hidden.
- maxlength: Indica el número máximo de caracteres que permite un campo de texto. Obviamente solo tiene sentido en controles de ese tipo.
- name: Define un nombre único para el elemento de entrada. Si se desea crear una pareja de control/valor al momento de enviar los los datos del formulario, es obligatorio para casi todos los tipos de controles excepto: submit y reset.
- readonly: Indica si el valor del control no puede ser modificado. Vñalido cuando el atributo type es igual a text.
- size: Define el tamaño del elemento de entrada. Vene definido en pixeles, a menos que el coontrol sea una entrada de texto. Para lo cual se define en caracteres.
- src: El URL de la imagen a presentar. Solo para tipos de control image.
- type: Indica el tipo de control del que se tratará. Valores posibles: button, checkbox, file, hidden, image, password, radio, reset, submit, text. La mayoría de estos tipos son auto explicativos y de fácil comprensión para cualquier usuario de un entorno gráfico de usuario. Los controles que pueden generar problemas de comprensión pueden ser: submit, image.- Ejecuta el contenido del atributo action del formulario, enviando su contenido. password.- Muestra un cuadro de texto que se enmascara al escribir.

reset.- Inicializa los controles del formulario a sus valores por defecto. hidden.- Especifica la existencia de un control invisible. file.- Crea un control de selección de archivo.

- value: Para controles de tipo botón (botton, submit, reset) define el texto del botón.

El elemento *button*

A pesar de existir tipos botón para el elemento <input>, existe por sí mismo un elemento <button> que cumple la misma función. Siendo más rico y flexible que los anteriores. Es un elemento de bloque.

Atributos

Botón <button>

- disabled: Especifica si el control está deshabilitado o no. Valor posible: disabled.
- name: Define un nombre único para el elemento de entrada.
- type: Indica de que tipo de botón se trata. Valores posibles: button, reset, submit. Realizan las isma funciones que sus homólogos del elemento <input>.
- value: Especifíca unvalor inicial para el botón. Dicho valor puede ser cambiado con un script.

El elemento *textarea*

Este control permite que el usuario introduzca texto de múltiples lineas. Es un elemento de línea.

Atributos

Área de texto <textarea>

- disabled: Especifica si el control está deshabilitado o no. Valor posible: disabled.
- name: Define un nombre único para el elemento.
- readonly: Indica que el contenido no puede ser modificado. Valor posible: readonly

Los elementos *select*, *option* y *optgroup*

Estos control permiten que el usuario seleccione una opción de entre una lista de ellas. Usualmente se representan como una lista desplegable.

El elemento <select> define la lista, el elemento <option> define cada una de las opciones de la lista, el elemento <optgroup> permite agrupar opciones. El elemento select es un

elemento de línea.

Atributos

Selección <select>

- disabled: Especifica si el control está deshabilitado o no. Valor posible: disabled.
- multiple: Indica que es posible realizar una selección múltiple. Valor posible multiple.
- name: Define un nombre único para el elemento de entrada.
- size: Especifica el número de elementos visibles de la lista.

Opción <option>

- disabled: Especifica si el control está deshabilitado o no. Valor posible: disabled.
- label: Define una etiqueta para usarse junto con optgroup.
- selected: Indica el elemento seleccionado por defecto. Valor posible: selected.
- value: Define el valor de la opción que será enviado al gestor de formularios.

Grupo de opciones <optgroup>

- disabled: Especifica si el grupo está deshabilitado pòr defecto. Valor posible: disabled.
- label: Define una etiqueta para el grupo de opciones.

El elemento *label*

El elemento label permite asociar explicitamente una etiqueta con un control para el cual no existe un atributo que se lo asocie directamente. Desde el punto de vista de un usuario de navegación visual tal vez no tenga mucho sentido, pero es muy útil a la hora de proporcionar un nombre a los controles a los que se accede desde un agente de usuario "no visual". Es un elemento de línea.

Atributos

Etiqueta <label>

- for: Indica el id del control al cual está asociado.

Los elementos *fieldset* y *legend*

El elemento fieldset permite agregar información estructural al formulario y agrupar controles. El elemento legend permite asociar una etiqueta al fieldset. El elemento fieldset es

un elemento de bloque.

Ninguno de estos elementos tiene atributos exclusivos.

Después de las explicaciones anteriores, un ejemplo:

Haz click aquí para ver el ejemplo 7.

También podemos incluir objetos dentro de nuestras páginas webs, tales como objetos Script y objetos Java, applets Java, pero eso también lo veremos más adelante, a lo largo del curso.

---

4. Elementos del XHTML

VIII. Objetos incrustados

Los objetos incrustados permiten introducir contenido que no es o no forma parte del HTML. En general, este contenido puede ser de cualquier tipo.

Los elementos que vamos a utilizar para ello son:

Elemento	Etiqueta	Es de tipo:
Objeto	<object>	bloque
Parámetro	<param>	--
Applet	<applet>	bloque

El elemento <applet> no forma parte de la especificación XHTML 1.0 Estricta, pero si de la versión Transicional. Hemos decidido incluirla por las siguientes razones:

- Es la manera más común de incrustar applets de Java.
- Algunos navegadores (lease *Internet Explorer*, por lo menos hasta la actual versión 6) no soportan de manera correcta el elemento <object>, que hace exactamente el mismo trabajo y aún mejor.
- Si no usáramos <applet>, ¡no habría manera de usar applets de Java en *Internet Explorer*!

Los elementos *object* y *param*

Este elemento ofrece la posibilidad (en teoría) de incrustar cualquier clase de objeto, como imagenes, video, sonido, aplicaciones, etc.

Los elementos <object> y <param> típicos se verán más o menos así:

```
<object classid="http://www.miamachina.it/relojanalogico.py">

 <param name="altura" value="40" valuetype="data" />
```

```
<param name="anchura" value="40" valuetype="data" />

 Este agente de usuario no puede representar aplicaciones Python.

</object>
```

En la etiqueta <object> se especifica el recurso a utilizar. En el ejemplo se trata de una aplicación Python. Si el recurso necesita el envío de parámetros, éstos pueden otorgarse por medio del elemento <param>. En el ejemplo se envían dos: anchura y altura.

Si el recurso no puede mostrarlo el navegador, se mostrará el contenido que se encuentre dentro de las etiquetas <object> y </object> como contenido alternativo. En el ejemplo ese contenido corresponde al mensaje: Este agente de usuario no puede representar aplicaciones Python.

Atributos

Los atributos exclusivos de estos elementos son los siguientes:

Objeto <input>

- archive: Una lista de URL's separadas con espacios. Las direcciones apuntan a archivos que contengan recursos externos para el recurso vínculado.
- classid: El identificador del recurso a incrustar. Puede ser un URL, o una entrada *classid* del registro de Windows, por ejemplo.
- codebase: Un URL que indica dónde se encuentra el código base del recurso.
- codetype: Indica el tipo MIME del contenido referido por classid.
- data: Define una URL con los datos del recurso.
- declare: Cuando está presente, este atributo booleano hace que la definición actual de <object> sea solamente una declaración. El objeto debe crearse por una definición <object> subsiguiente referida a esta declaración. Valor posible: declare.
- height: La altura del objeto en pixeles.
- name: Define un nombre único para el objeto.
- standbye: Especifica un texto que se muestra mientras el objeto se carga.
- type: Indica el tipo MIME del contenido referido por data.
- usemap: Indica un URL para un mapa de imagen del lado del cliente para ser usado con el objeto. define el texto del botón.
- width: El ancho del objeto en pixeles.

Parámetro <param>

- name: Define un nombre único para el objeto. Es obligatorio.

- type: Indica el tipo MIME del parámetro.
- value: Indica el valor del parámetro.
- valuetype: Indica el tipo MIME del valor. Valores posibles: data, ref, object.

El elemento *applet*

Este elemento permite incrustar aplicaciones Java en los documentos. Debe recordarse que este elemento no forma parte del estándar XHTML 1.0 Estricto. Está desaprobado en favor del elemento <object>.

Un elemento <applet> típico luce así:

```
<applet code="Burbujas.class" width="500" height="500">

Applet Java que dibuja burbujas animadas.

</applet>
```

Reformulado con <object> quedaría así:

```
<object codetype="application/java" classid="java:Burbujas.class" width="500" height="500

Applet Java que dibuja burbujas animadas.

</object>
```

Atributos

Los atributos exclusivos de este elemento son los siguientes:

Applet <applet>

- alt: Un texto alternativo para los agentes de usuario que no puedan representar el applet.
- archive: Una lista de URL's separadas con espacios. Las direcciones apuntan a archivos que contengan recursos externos para el recurso vínculado.
- code: Un URL que apunta al applet.
- codebase: Indica la ruta que servirá de base para el applet.
- name: Define un nombre único para el objeto.
- object: Defina el nombre de un recurso que contiene una representación serial del applet.
- height: La altura del objeto en pixeles.
- width: El ancho del objeto en pixeles.

Haz click aquí para ver el ejemplo 8.

4. Elementos del XHTML

IX. Mapas de imagen

Los mapas de imagen permiten dividir un gráfico en partes diferenciadas. Usualmente se utilizan para asignar un vínculo diferente a cada una de las partes.

Existen dos tipos de mapa de imagen: mapas del lado del servidor y mapas del lado del cliente. Para funcionar, los mapas del lado del servidor requieren el uso de lenguajes del lado del servidor. Este tipo de imágenes no los vamos a abordar en este Curso. Por el contrario, los mapas del lado del cliente permiten definir el mapa desde el código HTML. Veamos en qué consisten estos últimos.

Los elementos que debemos conocer para crear mapas del lado del cliente son los siguientes:

Elemento	Etiqueta	Es de tipo:
Mapa	<map>	(no se representa)
Área	<area>	(no se representa)

El elemento *map*

Este elemento permite definir un mapa de imagen particular. Dentro de él se especifican con el elemento <area> las áreas sensibles del gráfico.

Atributos

Los atributos exclusivos de este elemento son los siguientes:

- id: Una identificación únicsa para el elemento. En este caso es de carácter obligatorio.
- name: Cumple una función similar al anterior, y solo se mantiene por cuestiones de compatibilidad.

El elemento *area*

Este elemento permite definir cada una de las áreas en las que se dividirá la imágen.

Atributos

Los atributos exclusivos de este elemento son los siguientes:

- alt: Una descripción textual de área. Es de carácter obligatorio.
- shape: Especifica el tipo de figura que determinará el área. Los valores

posibles son: rect, rectangle, circ, circle, poly, polygon. Es de carácter obligarorio.

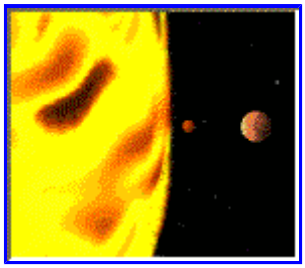
- href: Un URL que indica hacía donde apunta el vínculo.
- nohref: Especifica que el área en cuestión está excuida del mapa de imagen.
- coords: Permite definir las coordenadas que limitarán al área. Sus valores posibles dependen del atributo shape. A saber:
  - Si shape es rect o rectangle, la figura definida es un rectangulo. La sintaxis de coord es:
    coords="izquierda,arriba,derecha,abajo"
  - Si shape es circ o circle, la figura definida es un círculo. La sintaxis de coord es:
    coords="centro_x,centro_y,radio"
  - Si shape es poly o polygon, la figura definida es un polígono. La sintaxis de coord es:
    coords="x1,y1,x2,y2,..,xn,yn"

Trabajo conjunto

Los elementos <map> y <area> trabajan en conjunto. La manera en que lo consiguen es más o menos así:

```
<img src="imagenes/planetas.png" width="145" height="126" alt="Planetas" usemap="#plane
<map id="planetmap" name="planetmap">
 <area shape="rect" coords="0,0,82,126" href="son.html" alt="Sol" />
 <area shape="circle" coords="90,58,3" href="mercurio.html" alt="Mercurio" />
 <area shape ="circle" coords="124,58,8" href="venus.html" alt="Venus" />
</map>
```

Lo que tendría el siguiente resultado:



Como es posible apreciar, desde la etiqueta <img> se especifica qué mapa de imagen se utilizará con el atributo usemap. Dentro del elemento <map> se encuentran los elementos <area>.

---

5. Convertir HTML en XHTML manualmente

Convertir y comprobar que el documento XHTML es válido.

Son necesarios dos pasos para convertir un documento HTML en un documento XHTML válido.

- En primer lugar hemos de **asegurarnos** que está **bien formado**.
- Y en segundo lugar **ver que es válido** respecto de alguna de las tres DTD que conforman el XHTML.

Convirtiendo nuestro HTML en un documento XML bien formado

Simplemente tenemos que ir aplicando sobre el documento las reglas de buena formación del XML y eso lo podemos hacer utilizando las reglas que hemos definido en el apartado anterior.

Para ello, vamos a usar el siguiente ejemplo:

```
<html>
<script language="Javascript">
 function Hola(){
 alert ("Hola");
 }
</script>
<body>
<h1>Página HTML que convertiremos en XHTML</h1>
<p>Esto es un párrafo con una texto en <b>negrita <i> y</B> cursiva </i>.
<p>Esto es un párrafo en el que coloco un salto de linea. <br>
Y ahora continuamos aqui.</p>
<P align=center>Esto es texto centrado</P>
<ul>
 <li>Esto es una lista con un enlace
a una función de <a href="javascript:Hola()">javascript</a></li>
 <li>Esto es otra lista con una imagen <img src="lapiz.gif">
 <li><font size="4" color=red>Este es otro elemento de
la lista con otro color y tamaño de fuente mayor.</font>
</ul>
<p>Y para terminar un formularío</p>
<form>
<input type="checkbox" checked>Opcion 1
<input type="checkbox">Opción 2
```

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

Al que no tenemos más que aplicarle las reglas de las que hemos hablado en el apartado anterior. Lo mejor es empezar por poner todas las etiquetas en minúsculas, y luego dedicarnos a comprobar que todas las etiquetas se cierren correctamente, a entrecomillar los atributos, etc. Como ya hemos dicho en alguna ocasión, nuestro navegador habitual es una estupenda herramienta para comprobar que un documento XML esté bien formado, por ello igual es recomendabe convertir este documento HTML en un documento XML: añadir cabecera XML, etc e ir validando los cambios que vayamos haciendo en el navegador web.

Una vez realizados todos estos cambios y que hayamos conseguido verlos en el navegador web o validarlos respecto de un parser de XML, el documento HTML anterior quedaría de la siguiente manera:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<html>
<script language="Javascript" src="hola.js"></script>
<body>
<h1>Página HTML que convertiremos en XHTML</h1>
<p>Esto es un párrafo con una texto en <b>negrita </b> y <i>cursiva </i>.</p>
<p>Esto es un párrafo en el que coloco un salto de linea. <br />
Y ahora continuamos aqui.</p>
<p align="center">Esto es texto centrado</p>
<ul>
  <li>Esto es una lista con un enlace a
una función de <a href="javascript:Hola()">javascript</a></li>
  <li>Esto es otra lista con una imagen <img src="lapiz.gif" /></li>
  <li><font size="4" color="red">Este es otro elemento
de la lista con otro color y tamaño de fuente mayor.</font></li>
</ul>
<p>Y para terminar un formularío</p>
<form>
<input type="checkbox" checked="checked" />Opción 1
<input type="checkbox" />Opción 2
</form>
</body>
</html>
```

Observar como el código que iba dentro de la etiqueta script lo hemos colocado en un fichero externo hola.js. En este caso no hubiera sido necesario ya que no tiene ningún carácter que dé problemas, pero es recomendable acostumbrarnos a trabajar de esta manera.

Comprobar que el documento XHTML es válido

El siguiente paso es comprobar que se ajusta a alguna de las DTD que conforman el XHTML. En nuestro caso, al no utilizar CSS y darle el aspecto mediante etiquetas, tenemos

que comprobar que se ajusta a la transitional.dtd.

Para ello no tenemos más que colocar en la cabecera del documento la declaración de tipo de documento:

```
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "xhtml1-transitional.dtd">
```

Y al validar,lo primero que observamos es la obligatoriedad que exista el elemento head.

```
<!ELEMENT html (head, body)>
```

Y dentro de este,el elemento title

A continuación observamos cómo se queja que el elemento script debe utilizar de forma obligatoria el atributo type:

```
<!ATTLIST script
  charset    %Charset;     #IMPLIED
  type       %ContentType;  #REQUIRED
```

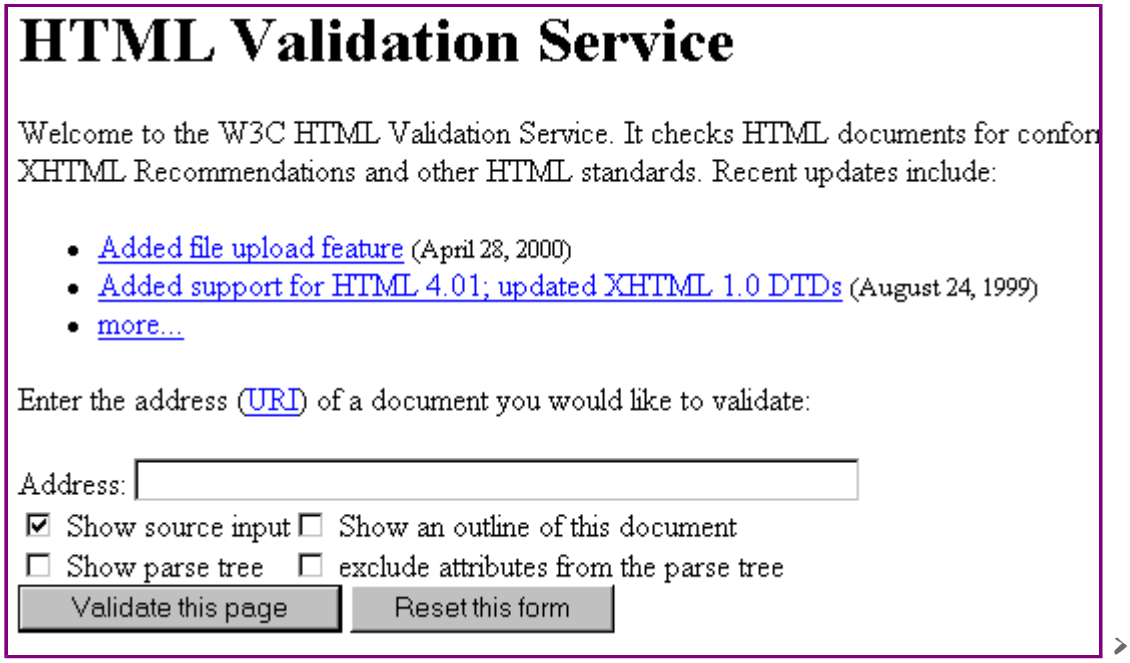
Donde debemos indicar de qué tipo es el contenido de la etiqueta script en nuestro caso text/javascript.

Y así sucesivamente vamos solucionando todos los problemas que nos dé el validador, hasta que obtengamos el siguiente documento XHTML válido:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "xhtml1-transitional.dtd">
<html>
<head>
<title>Ejemplo de documento HTML convertido a XHTML</title>
<script type="text/javascript" language="Javascript" src="hola.js"></script>
</head>
<body>
<h1>Página HTML que convertiremos en XHTML</h1>
<p>Esto es un párrafo con una texto en <b>negrita </b> y <i>cursiva </i>.</p>
<p>Esto es un párrafo en el que coloco un salto de linea. <br />
Y ahora continuamos aqui.</p>
<p align="center">Esto es texto centrado</p>
<ul>
  <li>Esto es una lista con un enlace a
una función de <a href="javascript:Hola()">javascript</a></li>
  <li>Esto es otra lista con una imagen
<img src="lapiz.gif" alt="Imagen de un lapiz" /></li>
```

```
 <li><font size="4" color="red">Este es otro elemento
de la lista con otro color y tamaño de fuente mayor.</font></li>
</ul>
<p>Y para terminar un formularío</p>
<form action="accion_asociada">
<input type="checkbox" checked="checked" />Opcion 1
<input type="checkbox" />Opción 2
</form>
</body>
</html>
```

También tenemos la posibilidad de validar nuestro código en Internet desde el validador que el W3C tiene en http://validator.w3.org/



Resumiendo

Por lo que hemos visto, no resulta muy complicado convertir un documento HTML a un documento XHTML válido. Depende esencialmente de lo mal que tengamos nuestro documento HTML. Si **corregimos nuestros malos hábitos en el momento de codificar HTML**, es decir,no nos olvidamos de entrecomillar todos los atributos, cerramos todas las etiquetas, las anidamos correctamente, mantenemos la estructura esencial de la página, etc., y lo escribimos todo en minúsculas, **la conversión de HTML a XHTML tendría que ser prácticamente inmediata**.

---

6. Convertir HTML en XHTML automáticamente

Instalación y funcionamiento de Tidy. Otras aplicaciones.

Es posible realizar la conversión de documentos HTML en XHTML de forma totalmente automática. Para ello deberemos utilizar una herramienta desarrollada por Dave Raggett denominada Tidy, que podemos obtener en el web del W3c.

Instalación y funcionamiento de Tidy.

Su instalación no tiene ningún inconveniente, ya que es un simple fichero .exe que no necesita ninguna instalación.

La ejecución se realiza desde la linea de comandos y debemos escribir:

tidy -f errores.txt fichero_entrada > fichero_salida

Con el parámetro -f indicamos el fichero donde se escribirán los errores del documento HTML. Para el documento HTML phtml.htm del apartado anterior escribiríamos:

tidy -f errores_phtml.txt phtml.htm > phtml_n.htm

Debemos observar que la salida no es XHTML, sino simplemente un documento HTML más correcto que el que nosotros habíamos escrito. Algunos errores son modificados por la aplicación y otros simplemente son indicados en el fichero de errores errores_phtml.txt para que nosotros los modifiquemos.

El **Tidy** dispone de un amplio número de parámetros que nos permiten parametrizar la salida de maneras muy diferentes. Para ver algunas de estas posibilidades no tenemos más que escribir:

tidy -h

y obtendremos:

```
Utility to clean up & pretty print html files
see http://www.w3.org/People/Raggett/tidy/
options for tidy released on 30th April 2000
  -config <file>  set options from config file
  -indent or -i   indent element content
  -omit   or -o   omit optional endtags
  -wrap 72        wrap text at column 72 (default is 68)
  -upper  or -u   force tags to upper case (default is lower)
  -clean  or -c   replace font, nobr & center tags by CSS
  -raw            leave chars > 128 unchanged upon output
  -ascii          use ASCII for output, Latin-1 for input
  -latin1         use Latin-1 for both input and output
  -iso2022        use ISO2022 for both input and output
  -utf8           use UTF-8 for both input and output
  -mac            use the Apple MacRoman character set
  -numeric or -n  output numeric rather than named entities
  -modify or -m   to modify original files
```

```
  -errors or -e   only show errors
  -quiet or -q    suppress nonessential output
  -f <file>       write errors to named <file>
  -xml            use this when input is wellformed xml
  -asxml          to convert html to wellformed xml
  -slides         to burst into slides on h2 elements
  -version or -v  show version
  -help   or -h   list command line options
Input/Output default to stdin/stdout respectively
Single letter options apart from -f may be combined
as in:  tidy -f errs.txt -imu foo.html
You can also use --blah for any config file option blah
For further info on HTML see http://www.w3.org/MarkUp
```

Hay que tener cuidado con la opción -m mediante la cual estamos indicando que los cambios se producen sobre el fichero original.

En la documentación de la aplicación se encuentran definidos con más detalle todos los parámetros que podemos utilizar. Entre ellos el parámetro --output-xhtml, mediante el cual indicamos que la salida sea XHTML.

Por tanto ,escribiendo:

```
tidy --output-xhtml yes --alt-text imagen phtml.htm > pxhtml_a.htm
```

Estamos convirtiendo el fichero phtml.htm en el fichero XHTML pxhtml_a.htm de forma automática, e incluso le estamos indicando que en las imágnes que no hemos puesto el atributo alt nos ponga por defecto el texto imagen.

NOTA: Observar que si validamos el fichero pxhtml.htm con nuestro parser de XML nos dará un error ya que en el elemento form no hemos indicado el atributo action que evidentemente hace falta. Si lo corregimos a mano tendremos un documento XHTML perfecto.

El **Tidy** es capaz de trabajar con un fichero de configuración,lo que nos permite no tener que reescribir los parámetros todas las veces que lo utilicemos. Este fichero de configuración se referencia mediante el parámetro: -config.

Por tanto, para el ejemplo anterior podríamos escribir el siguiente fichero tidyconf.txt

```
alt-text: imagen
output-xhtml: yes
```

Y ejecutando la aplicación de la siguiente manera obtendriamos los mismos resultados.

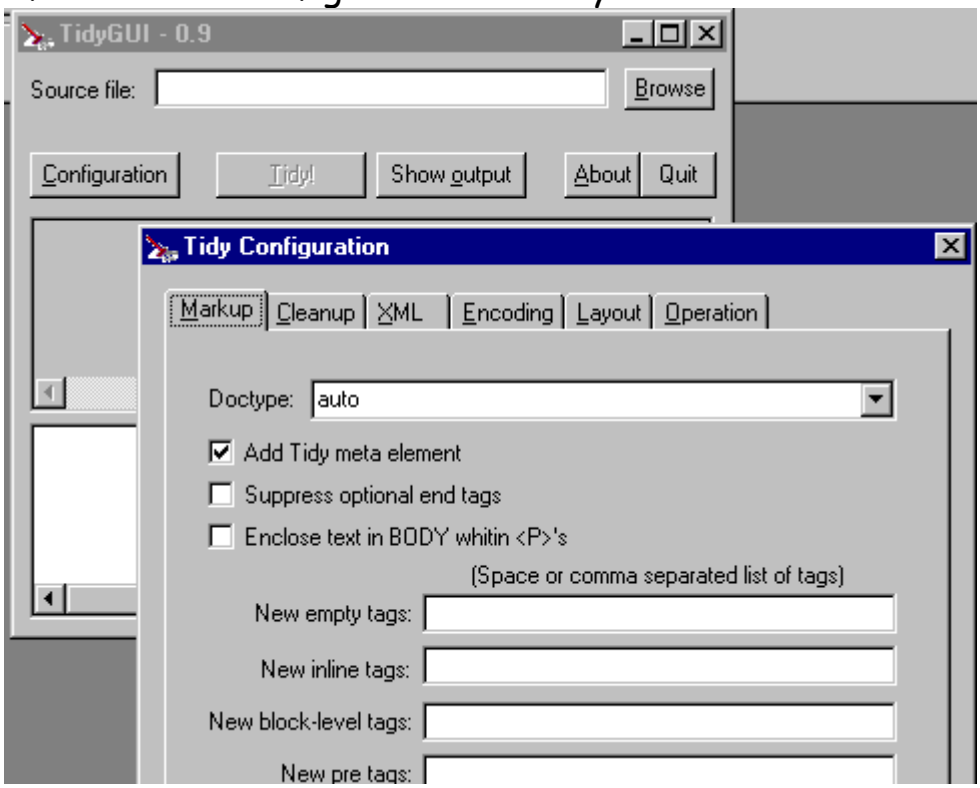
```
tidy -config tidyconf.txt phtml.htm > pxhtml_a.htm
```

Otras aplicaciones basadas en Tidy

La aplicación **Tidy** es cada vez más utilizada, en especial porque el uso del XHTML se está popularizando y la facilidad que nos ofrece para convertir nuestros documentos HTML en XHTML. Esto ha contribuido a que el número de aplicaciones que surgen alrededor de

Tidy sea cada vez mayor. Esta es una pequeña lista de estas aplicaciones:

- TidyGUI, es una aplicación de Windows escrita por André Blavier que permite la ejecución de Tidy desde un entorno de ventanas. Es muy recomendable y más si tenemos en cuenta que sólo ocupa 300K y no necesita ninguna insstalación. Resulta muy útil sobretodo para crear de forma los ficheros de configuración de Tidy.



- HTML-Kit, es un editor HTML gratuito para el entorno Windows que tiene integrado un soporte para Tidy.
- etc...

---

7. Visualizar XHTML en los navegadores actuales.

Como habéis podido observar,los documentos XHTML que hemos generado se visualizan perfectamente en vuestro navegador. Aunque esto no es así en todos los navegadores. Y en algunos de éllos hay problemas ,debido sobretodo a los elementos vacíos como
, <hr/>, etc.

Aunque, **según la especificación no hay ninguna obligación que los documentos XHTML 1.0 sean compatibles con los navegadores existentes, en la práctica es algo fácil de conseguir**. Las principales directrices para crear documentos compatibles según la especificación son las siguientes:

- **Elementos vacíos**. Incluir un espacio en blanco antes de la barra y ángulo

de cierre / y > de los elementos vacíos, por ejemplo:
, <hr /> y . También usar la sintaxis minimizada de etiquetas para los elementos vacíos, por ejemplo
, dado que la sintaxis alternativa a
</br> permitida por XML da resultados no previsibles en muchos de los navegadores ya existentes. Debemos fijarnos como la aplicación Tidy tiene en cuenta esta regla en el momento de generar el XHTML.

- **Minimización de elementos**. Dado un elemento vacío cuyo modelo de contenido no es empty, como por ejempo un título o un párrafo, no utilizar la forma minimizada, es decir escribirlo como: en lugar de .
- **Hojas de estilo y archivos de código incrustados**.Usar hojas de estilo externas o ficheros de código externo si la hoja o el código en cuestión utiliza los caracteres < o & o]]> o --. Notar que los analizadores XML tienen permitido suprimir el contenido de los comentarios. De esta manera, la práctica común hasta ahora de "esconder" los fragmentos de código (script) y hojas de estilo (style) entre comentarios, para hacerlos invisibles a antiguos navegadores, normalmente no funcionará en aplicaciones basadas en XML.
- **Saltos de línea dentro de valores de atributos**. Evitar saltos de línea y múltiples espacios en blanco dentro de los valores de los atributos. Estos son manipulados de manera inconsistente por los navegadores.
- **Identificadores de fragmentos**. En XML, los URI que terminan con identificadores de fragmentos del tipo #identificador no se refieren a elementos con un atributo name=identificador, sino que se refieren a elementos con un atributo de tipo ID. Muchos navegadores actuales no soportan este uso de atributos de tipo ID, de tal manera que se pueden dar valores idénticos a ambos atributos para asegurar la máxima compatibilidad futura y retroactiva. Por tanto, en estos casos es recomendable escribir: .... Finalmente, notar que XHTML 1.0 tiende a desechar el atributo name de los elementos.
- **Uso del carácter & en valores de atributos**. Cuando el valor de un atributo contenga un carácter &, debe expresarse como una referencia a la entidad de tipo carácter (por ejemplo: &"). Por ejemplo, cuando el atributo href del elemento a apunte a un código CGI que tome parámetros debe expresarse como http://www.ciberaula.com/cgi-bin/programa.pl?dato1=33&dato2=43 en lugar de http://www.ciberaula.com/cgi-bin/programa.pl?dato1=33&dato2=43

- **Codificación de caracteres**. Para especificar una codificación de caracteres en el documento, usar tanto la especificación del atributo de codificación en la declaración XML (por ejemplo <?xml version="1.0" encoding="iso-88590-1"?>) como una sentencia meta http-equiv (por ejemplo <meta http-equiv="Content-type" content='text/html; charset="iso-88590-1"' />). El valor del atributo de codificación de la instrucción de proceso XML tiene preferencia.

Existen algunas directrices más, pero siguiendo ésta tendría que ser suficiente para conseguir que nuestros documentos XHTML se vean correctamente en los navegadores actuales.

Los profesores de este Curso WebCSS son:
A. Gámez, L.M. Marín, F. Mesa & S. Fandiño
