Tutorial de HTML

HTML y CSS

Juan Ramón Pozo

http://conclase.net

Empezando desde cero: ¿Cómo funciona la Web? (I)

Introducción

- Bien, así que quieres aprender a hacer páginas web, ¿no?

- Pues... sí, básicamente por eso estoy aquí.

- Claro. Supongo también que ya estás familiarizado con la Web. Sabes navegar y todo eso.

- Sí. Sí, claro...

- Claro, si no no estarías aquí. Y, ¿cómo llegaste hasta esta página, si puedo preguntar?

- Ee... pues, ya sabes, llegué desde otra página, pinchando en un enlace...

- ¡Bien! A partir de ahora, quiero que recuerdes una cosa: **la Web son documentos enlazados unos con otros**. Grabátelo en el coco, ¿lo harás?

- Bueno, si insistes... Pero ¿tan importante es?

- Sí, quiero que construyas tus conocimientos en bases sólidas, así que es fundamental que sepas cuál es la esencia de la Web.

- Muy bien, la Web es un montón de documentos enlazados los unos con los otros.

- Estupendo. Por supuesto tiene que haber algún mecanismo que haga eso posible.

- ¿A qué te refieres exactamente con un mecanismo?

- Pues, piensa en todo lo que tiene que suceder cada vez que tú haces algo tan fácil como pinchar un vínculo con el ratón. Cuando

haces clic, tu ordenador recibe la orden de buscar el documento que le has pedido. Así que tiene que ir hasta donde esté el documento en cuestión, recogerlo, y mostrártelo en la pantalla.

- Sí, a grandes rasgos eso es lo que sucede.

- Bien. Pongámonos en el lugar del ordenador. Lo primero que necesitamos es saber qué documento tenemos que ir a buscar...

- Obviamente...

- Y necesitamos sabér 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.

- Vaya, pues es algo más complicado de lo que parecía a simple vista.

- Bueno, desde el punto de vista técnico hay muchas operaciones, que en realidad no deben preocuparte, pero te cuento todo esto porque desde el punto de vista del diseñador esto es importante. De aquí salen algunos conceptos que son fundamentales.

- Soy todo oídos.

- 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 computadora de todas las conectadas a Internet está almacenado. En segundo lugar, debe existir un lenguaje en que las computadoras se comuniquen para pedirse y entregarse documentos unas a otras. Y en tercer lugar, debe existir una forma de codificar los documentos para que una vez que un ordenador, sea el que sea, lo haya obtenido, sea capaz de representarlo en la pantalla o en cualquier otro medio.

- Si quieres que te sea sincero, creo que estoy empezando a perderme...

- Tranquilo, es normal. Vamos por partes.

- Sí, será lo mejor.

Buscando documentos por la red

- Muy bien. Hablemos del localizador.

- Sí, eso es fácil. Cada documento necesita un nombre único que le diferencie de todos los demás.

- Exacto. Y ya estarás harto de verlo: es lo que pone en la barra de direcciones del navegador. Por ejemplo, el identificador de esta página es `http://html.conclase.net/tutorial/html/1/2`.

- Entiendo.

- Ya veremos con más detalle qué significan exactamente todas las partes de un identificador. De momento lo importante es que si por ejemplo quieres crear un documento que esté enlazado con éste, tienes que hacer referencia a él con su nombre, `http://html.conclase.net/tutorial/html/1/2`.

- Muy bien, creo que lo capto, aunque sigo un poco perdido.

- Sigamos. La segunda parte era la comunicación entre computadoras.

- ¿Y de verdad yo me tengo que meter en eso? Me da un poco de miedo...

- No, al menos no hasta que tengas un nivel más avanzado. Pero esto también lo has visto, seguro. ¿Alguna vez te ha salido en la pantalla eso de "Error 404: Not found" o algo por el estilo?

- ¡Sí, ya te digo! ¡Un montón de veces!

- Pues ahí lo tienes. Las computadoras 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. Lárgate".

- ¡Qué amable!

- Ya ves. Por supuesto, como sabrás, los ordenadores hablan con números, y en vez de decir "No lo encuentro" dicen "404". Así que tu ordenador vuelve derrotado con el rabo entre las piernas y un "404" en sus manos...

- ¡Y no se le ocurre nada mejor que darme el 404 a mí!

- 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.

- HTTP, como lo que sale en el identificador...

- Justo. Ya veremos por qué, más adelante.

- ¿Qué significa?

- HTTP es HyperText Transfer Protocol.

- Esto... me dijeron que este tutorial era en español...

- Sí, sí. HTTP: Protocolo para la Transferencia de Hipertexto.

- Pues... bueno, no hemos mejorado mucho, la verdad...

- Bueno, lo de Protocolo de Transferencia seguro que ya lo has pillado.

- Sí, eso ya lo pillo.

- Y lo de hipertexto...

- Sí, eso es muy cíber...

- Compórtate, hombre.

- Perdón...

- Hiper es un prefijo que significa...

- Grande, ¿no? Cómo "hipermercado", por ejemplo.

- ¿Me vas a dejar? Hiper es un prefijo que usan los matemáticos para indicar que algo tiene dimensiones extras.

- ¿Cómo en hiperespacio?

- Sí, ese es un ejemplo. El hiperespacio tiene más de tres dimensiones. El texto normal tiene una sola dimensión...

- De eso nada, los documentos tienen dos dimensiones: mira una hoja de papel, es plana.

- No. Un cuadro, tiene dos dimensiones. Un texto tiene una dimensión, es como una línea. Empiezas a leer desde el principio y acabas por el final. Si pierdes el hilo deja de tener significado.

- Sí, claro, visto así...

- Entonces hipertexto es como el texto normal, pero mejor: puedes salirte del hilo, porque no estás restringido a una sola dimensión.

- ¡Claro, puedes ir dando saltos, como en la Web!

- ¿Has visto? ¡Todo tiene su explicación!

- ¡Y yo que pensaba que hipertexto sólo era un nombre chulo! Pero oye, no veo cómo esto del protocolo HTTP nos puede ser útil como diseñadores.

- Pues, por ejemplo, si controlas el protocolo, le puedes decir a la computadora en que están tus documentos que sea más amable, y que en vez de dar un 404, entregue una página más bonita, explicando que la página no ha sido encontrada, dando las posibles causas, una dirección de contacto y una forma de llegar a una página que exista.

- Aah, claro... pues es verdad. Bueno, vamos a la tercera parte, ¿no?

- ¡Ah, sí! Ya se me había olvidado...

¿Qué es el HTML?

- 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.

- Ya te estás poniendo críptico otra vez.

- Perdón. Voy a tratar de explicártelo. Si los documentos fueran de texto normal, bastaría con enviar un fichero con las palabras del texto, un documento de texto normal y corriente.

- Sí, pero no es texto normal...

- 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.

- Mmm... ya, creo que te entiendo. Y por casualidad, ¿ese lenguaje es el HTML?

- Exactamente. El HTML fue 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, ya lo verás.

- Eso espero, porque esta lección me está asustando un poco, hablas muy raro.

- Ya te acostumbrarás. Además ya sabes que los comienzos siempre son más difíciles que el resto.

- Sí. Bueno, ¿y qué significa HTML?

- HyperText Mark-up Language

- Ya estamos otra vez...

- Lenguaje para el Formato de Documentos de Hipertexto. Más claro el agua.

- ¿Quieres decir que Mark-up significa Formato de Documentos?

- Pues, 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 "mark-up".

- Interesante.

- Pues eso es justamente lo mismo que tu vas a hacer cuando escribas en HTML. Tu vas a ser a la vez el escritor (escribes el contenido del documento) y el editor, porque vas 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. Y el navegador será como la imprenta, que va a reconocer todas esas marcas y le va a dar a tu documento la apariencia deseada: los títulos más grandes, los párrafos separados, las listas con marcadores de lista, etc.

- Ya entiendo. La idea en sí parece sencilla, ¿no?

- Sí, ¿quieres ver un ejemplo?

- ¡Claro!

Un ejemplo de HTML

- Muy bien. Imagina que tu página web va a tratar de... ¿de qué quieres que trate?

- No sé, por ejemplo, podría poner mis discos de música.

- Buena idea. Podríamos hacer un documento con un título como "Mi colección de discos", y varias secciones donde hablaras un poco de cada grupo y pusieras una lista con los discos. Dime un par de grupos.

- Pues, por ejemplo, Horslips y Gwendal.

- ¿Eh? Bueno, da igual. El caso es que el documento podría tener esta estructura:

```
Mi colección de discos

Horslips
Aquí ponemos algo sobre la historia de este grupo.
Después ponemos la lista de discos:
- El primer disco
- El segundo disco
- etc.

Gwendal
Lo mismo, un poco sobre el grupo y luego la lista:
- El primer disco
- El segundo disco
- etc.
```

- Ya veo. Esto es como un boceto. ¿Qué haríamos ahora?

- Si recuerdas, te dije que haríamos la labor del escritor y del editor. Además de decir lo que vamos a poner, que es lo que hace el escritor, tenemos que marcar la estructura del documento: los títulos, los párrafos, las listas...

- Justo lo que tenemos aquí: títulos, párrafos y listas.

- Muy bien, pues vamos a ello. Le vamos a poner a cada cosa, a cada elemento del documento, su etiqueta, para marcar su función dentro del documento, dónde empieza y dónde acaba. Por ejemplo, a los párrafos les ponemos una etiqueta `<p>` al principio, y otra `</p>` al final, donde `p` significa "párrafo". Al título principal, una etiqueta `<h1>` al principio y otra `</h1>` al final, y así todo.

- Pues no parece muy complicado...

- Claro, es que no lo es. Así es como quedaría el ejemplo completo. No te preocupes por las etiquetas que no conoces, ya hablaremos de ellas más adelante:

```
<h1>Mi colección de discos</h1>

<h2>Horslips</h2>
<p>Aquí ponemos algo sobre la historia de este grupo.
Después ponemos la lista de discos:</p>
```

```
<ol>
<li>- El primer disco</li>
<li>- El segundo disco</li>
<li>- etc.</li>
</ol>
<h2>Gwendal</h2>
<p>Lo mismo, un poco sobre el grupo
y luego la lista:</p>
<ol>
<li>- El primer disco</li>
<li>- El segundo disco</li>
<li>- etc.</li>
</ol>
```

- Ajá, supongo que `<h2>` son títulos de segundo nivel, `<ol>` listas y `<li>` cada objeto de la lista, ¿no?

- Muy bien, ¿has visto? ¿A que es fácil? Ahora sólo tienes que escribir el texto real y podemos ver cómo queda.

- Vale, a ver qué tal se ve esto:

```
<h1>Mi colección de discos</h1>

<h2>Horslips</h2>
<p>Este grupo irlandés nació en 1970, y en sus diez años de
existencia, hasta su separación en 1980, recorrieron Europa
y América tocando en directo. Aunque su música evolucionó
sensiblemente en ese tiempo, siempre se caracterizó por
imprimir de manera única las raíces celtas y el sonido del
rock. Horslips, sin haber llegado a ser un fenómeno de
masas, ha escrito algunas de las páginas más brillantes de
la música irlandesa reciente.</p>
<ol>
<li>"Happy To Meet, Sorry To Part", 1972</li>
<li>"The Táin", 1973</li>
<li>Etcétera, no los pongo todos.</li>
</ol>
<h2>Gwendal</h2>
<p>Gwendal es probablemente uno de los grupos de música celta
más conocidos en España. No en vano, su único disco en
directo fue grabado en España. Este grupo bretón, da a los
temas tradicionales irlandeses un estilo único, mezclando
delicadeza, virtuosismo y energía de manera magistral.</p>
<ol>
<li>- "Gwendal", 1974</li>
<li>- "Gwendal 2", 1975</li>
```

```
<li>Etcétera, tampoco los pongo todos</li>
</ol>
```

- No te ofendas, pero como crítico musical te queda bastante por aprender. Bien, le he añadido una línea que es necesaria pero que no influye en la presentación. Ya hablaremos de eso en el futuro. Pero a ver cómo queda.

Las hojas de estilo

- Estooo, no te ofendas, pero como tutor de HTML te queda bastante por aprender. Esto es horrible. Sería más bonito si lo escribiera a mano. No pretenderás hacerme creer que todas esas páginas web tan estupendas que se ven por ahí están hechas así...

- ¡Eeh, tranqui! Muchacho, si acabamos de empezar, ¿qué quieres? De todos modos tienes algo de razón. El HTML no sirve para hacer cosas bonitas. Puedes estructurar tus contenidos, pero poco más...

- ¿Entonces?

- Verás, el HTML tiene algo así como unas "amigas" que le ayudan a hacer lo que él no sabe.

- Me dejas sin palabras. ¿Qué clase de amigas?

- Las Hojas de Estilo. Especialmente las CSS.

- No me lo digas, más palabros ingleses.

- Pues sí. CSS significa "Cascading Style Sheets"

- Eso suena muy mal, tío.

- "Hojas de Estilo en Cascada". Sería un poco largo explicarte por qué se llama así. Así que de momento quédate con lo de "Estilo".

- Muy bien. Veamos, supongo que las hojas de estilo nos dejan darle estilo al HTML, ¿no?

- Exactamente. Voy a intentar explicárte de manera sencilla cómo funciona esto.

- Más te vale.

- Tenemos nuestro cutre documento en HTML y nos apetece darle un poco de color al asunto. Para ello, tenemos que escribir unas "reglas de estilo" para el navegador. Si todas esas reglas las juntamos en otro documento, lo que tenemos es una hoja de estilo.

- Muy bien, sigue...

- Ya está, esa es la idea básica. Pero es una idea muy básica y muy fundamental del diseño web: **el contenido y su estructura en HTML, el estilo y la apariencia en la Hoja de Estilo**. ¿Lo recordarás?

- ¡Por supuesto! ¿Por quién me has tomado?

- Genial. Te pondría un ejemplo, pero creo que para ser el primer día hemos avanzado suficiente. ¿Qué has aprendido hoy?

- He aprendido que la Web son documentos enlazados entre sí. Que eso es posible gracias al hipertexto, que te permite saltar de unos documentos a otros. Y también gracias a que cada documento tiene un nombre propio que lo identifica entre los demás. Que los documentos de hipertexto viajan por la red a través de un protocolo llamado HTTP. Que para escribir documentos de hipertexto se usa HTML. Y que en los documentos HTML se mete el contenido y su estructura, y la apariencia se controla con las Hojas de Estilo.

- Muy bien, aunque hay algo incorrecto en lo que has dicho, y esa es la tercera y última regla fundamental que te voy a dar hoy: **en la Web la apariencia no se controla**. Aunque quisieras no podrías. No sabes quién lee tus páginas, no sabes que clase de ordenador utiliza, ni el navegador que utiliza, ni el sistema operativo, ni la resolución de pantalla, ni el número de colores, ni el tamaño de la ventana de su navegador... Por tanto, recuerda, en la Web no controlas, sólo sugieres.

- De acuerdo.

- Y es importante que te tomes esto como una ventaja de la Web, y no como un inconveniente contra el que luchar. Gracias a su enorme flexibilidad, la Web, y con ella lo que tú escribas, puede ser universal. Si escribes tus documentos correctamente, hasta los ciegos podrán acceder a sus contenidos sin dificultad.

- Muy bien, muy bien, me has convencido: en la Web no se controla nada, sólo se sugiere.

- Eres un alumno estupendo. Espero verte por aquí pronto para la segunda lección. Tengo algunas cosas muy importantes que enseñarte.

- Por supuesto que sí, no faltaré

- Hasta luego.

- Adiós, ¡y gracias!

La estructura de un documento HTML (I)

Introducción

- Hola, ¿qué tal? ¿Preparado para la siguiente lección?

- Sí, pero antes tengo un par de preguntas sobre el ejemplo del otro día.

- A ver...

- En primer lugar, las líneas de los párrafos no se partían por donde nosotros escribimos.

- Lógico. Pensé que eso lo habías entendido. Recuerda siempre que un párrafo es un párrafo. Comienza con `<p>` y termina con `</p>`. Todo lo que haya entre esas dos etiquetas es un único párrafo. Aunque insertes cien líneas en blanco entre cada dos palabras, seguirán formando un único párrafo. El navegador lo representará como tal. En el caso de un navegador visual, cuanto más estrecha sea la ventana, más alto será el párrafo.

- Ya veo...

- Es la última regla básica que te di en el primer tutorial: no controlas la apariencia.

- ¿Entonces cómo hago para que haya un salto de línea donde yo quiera?

- Muy sencillo: el párrafo termina donde existe un salto de línea.

- Claro, visto así...

- ¿Tenías otra pregunta?

- Sí, me fijé en el fuente del ejemplo, y vi que habías añadido una línea así al principio: `<title>Primer ejemplo en HTML</title>`

- En efecto. En HTML cada documento debe tener un título, que viene especificado por el elemento `TITLE`. Lo añadí porque es obligatorio.

- Pero el documento ya tenía un título, ¿te acuerdas? Lo pusimos así: `<h1>Mi colección de discos</h1>`. ¿No vale con eso?

- No, es diferente. `TITLE` se refiere al título del recurso, del fichero `.html` si lo entiendes mejor así. `H1` es el título de una sección de los contenidos del cuerpo del documento. Puede haber más de un `H1` en un mismo documento. Lo entenderás mucho mejor al final de este tutorial.

- ¿De qué vamos a hablar hoy?

- Había pensado en hablarte de la estructura de un documento HTML.

- Muy bien, espero que no te pongas muy abstracto.

- Lo intentaré. También había pensado que al final podrías crear tu propio documento HTML.

- ¡Eh, eso sería genial! Pero me hará falta algún programa, ¿no?

- Nada que no tengas ya: un navegador para ver la página y el editor de textos más sencillo que haya en tu sistema operativo. Si estás en Windows, el Bloc de Notas será perfecto para empezar.

- ¿En serio?

- Completamente. ¿Empezamos?

- ¡Empecemos!

Elementos y su estructura

- Veamos cómo te explico esto sin liarme mucho. Un documento HTML está formado por *elementos*. Ya hemos hablado de elementos. Por ejemplo, un elemento `P` representa un párrafo, un elemento `OL` representa una lista, un elemento `TITLE` representa el título del documento...

- Sí, todo eso ya lo hemos visto.

- Bien. En general, cada elemento se divide en tres partes: una *etiqueta inicial*, el *contenido del elemento*, y una *etiqueta final*.

- Como por ejemplo `<p>Esto es un párrafo.</p>`, ¿verdad?

- Exacto. Y nunca confundas etiqueta con elemento. **Un elemento no es una etiqueta**. Una etiqueta es una parte de un elemento. Si los confundes todo el mundo sabrá que no sabes de qué estás hablando.

- Bueno, bueno, vale.

- Unos elementos pueden contener a otros elementos. Por ejemplo, en el caso de nuestro ejemplo, el elemento `OL` contenía elementos `LI`. El elemento `OL` representa una lista, y cada elemento `LI` representa uno de los elementos de la lista.

- ¿No podríamos haber metido los elementos `LI` en el elemento `P`?

- No, existen reglas que definen qué tipos de elementos puede contener cada tipo de elemento. Por ejemplo, un elemento `P` no puede contener a otros elementos `P`.

- Claro, es bastante lógico...

- Otro ejemplo, un elemento `OL` sólo puede contener elementos `LI`. Así es como está definido. Cada tipo de elemento tiene sus reglas.

- ¿Y hay muchos tipos de elemento?

- Sí, pero no te preocupes, normalmente son reglas lógicas. El HTML es realmente sencillo. Pero a lo que íbamos.

- Sí, sí, la estructura del documento...

- Pues bien, un documento HTML sólo puede contener un tipo de elemento: el elemento `HTML`.

- ¿Hein?

- Como lo oyes. Es como si el documento fuera un gran elemento. A su vez, el tipo de elemento `HTML` puede contener dos tipos de elemento: un elemento `HEAD` y un elemento `BODY`, y sólo uno de cada. El elemento `HEAD` define la *cabecera* del documento, que contiene información sobre el documento, y el elemento `BODY` contiene el *cuerpo* del documento, lo que será representado por el navegador.

- Eso no me cuadra. En nuestro ejemplo no había nada de eso.

- Sí lo había, pero no lo veías.

- ¿Eh? ¿Qué dices? No te entiendo nada.

- Para algunos tipos de elementos, las etiquetas son opcionales. Para otros tipos, sólo la etiqueta final es opcional. Pero eso no quiere decir que el elemento no empiece y no termine. Lo primero que hace el analizador del código es buscar dónde empieza el elemento `HTML`. Si no encuentra la etiqueta inicial, supone dónde está.

- Un poco complicado, ¿no?

- En principio no, pero se puede complicar bastante. Por eso yo te recomiendo que incluyas siempre todas las etiquetas aunque sean opcionales. Además, si en el futuro haces tus páginas en XHTML tendrás que incluir todas las etiquetas, porque allí no hay etiquetas opcionales.

- Muy bien, te haré caso, así lo haré.

- He vuelto a escribir el ejemplo del primer tutorial, pero con todas las etiquetas explícitas:

```
<html>
 <head>
  <title>Primer ejemplo en HTML</title>
 </head>

 <body>
  <h1>Mi colección de discos</h1>

  <h2>Horslips</h2>
  <p>Este grupo irlandés nació en 1970, y en sus diez años de
existencia, hasta su
  separación en 1980, recorrieron Europa y América tocando en
directo. Aunque su
  música evolucionó sensiblemente en ese tiempo, siempre se
caracterizó por imprimir
  de manera única las raíces celtas y el sonido del rock.
Horslips, sin haber
  llegado a ser un fenómeno de masas, ha escrito algunas de
las páginas más
  brillantes de la música irlandesa reciente.</p>
  <ol>
   <li>"Happy To Meet, Sorry To Part", 1972</li>
   <li>"The Táin", 1973</li>
   <li>Etcétera, no los pongo todos.</li>
  </ol>
  <h2>Gwendal</h2>
  <p>Gwendal es probablemente uno de los grupos de música
celta más conocidos en
  España. No en vano, su único disco en directo fue grabado
en España. Este grupo
  bretón, da a los temas tradicionales irlandeses un estilo
único, mezclando
  delicadeza, virtuosismo y energía de manera magistral.</p>
  <ol>
   <li>"Gwendal", 1974</li>
   <li>"Gwendal 2", 1975</li>
   <li>Etcétera, tampoco los pongo todos</li>
  </ol>
 </body>
</html>
```

Cómo ves el elemento TITLE está contenido en HEAD, y todos los demás están contenidos en BODY.

- ¿Y no sería más lógico que los elementos `P` estuvieran contenidos en los elementos `H`?

- Probablemente sí, pero así es como está definido... Bueno, este documento es perfectamente válido, es decir, sigue todas las reglas sintácticas de HTML, pero normalmente los documentos son más complejos, y es fácil equivocarse. Por eso hay servicios que nos permiten validar nuestros documentos. Pero para eso hace falta un pequeño detalle más.

Validación y documentos correctos

- Normalmente el programa validador necesita conocer cuáles son las reglas que se aplican a nuestro documento, para saber si hemos aplicado correctamente esas reglas o no.

- ¿Y que hay que hacer para que lo sepa?

- Una manera de hacerlo es añadir al principio una *declaración del tipo del documento*. Nosotros estamos aprendiendo HTML 4.01, por tanto la declaración tiene que tener esta pinta:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

- ¡Qué cosas más raras!

- Sí, es que en realidad esto no es HTML, pero no te preocupes mucho. Lo ponemos y ya está. Ahora lo validamos, por ejemplo con el validador del W3C. Si todo va bien, no encontrará ningún error.

- A ver.

- Perfecto: *This page is valid.* Ahora viene un tema que es un poco polémico.

- ¿Cuál?

- Verás, según la especificación de HTML 4.01, es obligatorio incluir la declaración del tipo de documento en cualquier documento

HTML.

- ¿Y eso es polémico?

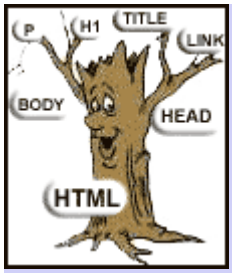
- Sí. Bueno, no debería serlo, pero lo es. El caso es que, nosotros vamos a incluirlo siempre, ¿de acuerdo?

- De acuerdo.

- Por cierto, si te fijas en los resultados del validador, verás que hay un apartado llamado *Parse Tree*. *Parse* significa *analizar*, y *tree* singnifica *árbol*.

- Ya lo veo.

- Normalmente se dice que la estructura de un documento HTML tiene forma de árbol. Hay un tronco que se ramifica, y de las ramas a su vez salen otras ramas, y así sucesivamente. Por ejemplo, el elemento `HTML` tiene dos ramas: `HEAD` y `BODY`:



Un parse tree simpaticote

- Interesante.

- Una cosa más acerca de la validación de documentos. El hecho de que un documento sea sintácticamente correcto, no quiere decir que sea un documento válido desde todos los puntos de vista.

- ¿Por ejemplo?

- Por ejemplo, hay gente que utiliza elementos `P` sin contenido para dejar espacio en blanco. Lo cual es completamente absurdo, porque es como si escribieras un párrafo sin palabras.

- Claro, entonces es un párrafo que no es un párrafo. No sería un párrafo...

- Para dejar espacio en blanco, se usan las hojas de estilo, porque eso no tiene que ver con la estructura, tiene que ver con la apariencia. Otro ejemplo. Hay gente que para conseguir escribir algo

en letras grandes y en negrita utilizan un elemento `H1`. En realidad, eso es incorrecto.

- Claro, porque `H1` contiene el título de una sección de un documento.

- Justamente. Para especificar que algo va en negrita, se puede usar el tipo de elemento `STRONG`, que significa "énfasis fuerte", y para que sea de tamaño mayor que el de las palabras que lo rodean, se usan hojas de estilo. Y así son comunes muchos truquitos igual de incorrectos.

- ¡Claro, pero todo eso hay que saberlo antes!

- Sí, es verdad. Pero antes me gustaría explicarte un par de cositas.

Elementos en bloque y en línea. Atributos

- En HTML, algunos de los tipos de elementos que pueden aparecer en `BODY` se pueden dividir en dos grandes grupos: *elementos en bloque* (o a nivel de bloque) y *elementos en línea* (o a nivel de texto).

- ¿Cuál es la diferencia?

- Se diferencian especialmente en dos aspectos. En primer lugar, normalmente los elementos en bloque pueden contener elementos en línea y algunos de ellos también a otros elementos en bloque. Los elementos en línea no pueden contener elementos en bloque. En segundo lugar, los elementos en bloque suelen provocar un salto de línea antes y otro después de los contenidos del elemento. Los elementos en línea no suelen provocar un salto de línea.

- Dame algún ejemplo, ¿no?

- Sí, por ejemplo, los elementos `P` (párrafo), `OL` (lista ordenada) y `LI` (objeto de lista) son elementos en bloque. Los elementos `EM`

(énfasis) y `STRONG` (énfasis fuerte) son elementos en línea. Un elemento `P` puede contener elementos `EM` y `STRONG` pero no puede contener elementos en bloque. El elemento `OL` sólo puede contener elementos `LI`. El elemento `LI` puede contener elementos en bloque y en línea.

- Muy bien, pero, ¿para qué me explicas todo esto?

- Porque es útil para describir qué elementos puede contener cada tipo de elemento.

- Ah, pues, ahora que lo dices, es verdad...

- La otra cosa que quería explicarte eran los *atributos* de los elementos. Verás, cada elemento tiene algunos atributos cuyo valor puedes asignar. Por ejemplo, para crear un hipervínculo normalmente se emplea el elemento `A`...

- ¿De dónde viene esa `A`?

- Es la inicial de *ancla* (en inglés *anchor*). El caso es que este elemento tiene una doble función: puede ser el *origen* de un hipervínculo, o el *destino* de un hipervínculo. Para que el elemento sea el origen de un hipervínculo, hay que darle un valor a su atributo `href`, y para que sea el destino de un hipervínculo, hay que darle un valor a su atributo `name`.

- ¿Y puede ser a la vez origen y destino?

- Sí.

- ¿Y cómo se asigna un valor a los atributos?

- Es muy sencillo. En la etiqueta inicial, después del nombre del elemento, incluyes una pareja `atributo="valor"` para cada atributo. Conviene que encierres siempre el valor entre comillas. Por ejemplo: `<a href="http://html.conclase.net/tutorial/html/index">¡Sigue nuestros tutoriales!</a>` aparece en tu navegador así ¡Sigue nuestros tutoriales!. Si lo pulsas, irás al índice del tutorial de HTML.

- Es cierto.

- Muy bien, en el tutorial siguiente hablaremos más a fondo sobre hipervínculos. Hay dos atributos muy importantes para añadir estilos a los documentos, que son `class` e `id`. También hablaremos sobre ello dentro de algunos tutoriales. De momento, ahora que ya sabes incluso cómo se hace un hipervínculo a otra página, es el momento de que crees tu propia página desde cero.

- ¡Qué emocionante!

Creación de un documento HTML

- Lo primero que tienes que hacer es abrir el editor de textos más tonto y piojoso que tengas instalado. Si estás en Windows, abre el Bloc de Notas. Estará en Inicio/Programas/Accesorios o algo por el estilo.

- Muy bien, hecho.

- Ahora escribe la declaración del tipo de documento. O si lo prefieres, copia y pega, es más sencillo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html40/strict.dtd">
```

Tiene que ser exactamente así.

- Ah, muy bien, muchas gracias. Ya está.

- ¿Ahora qué va?

- El elemento `HTML`, ¿no?

- Justo. Y dentro del elemento `HTML`...

- `HEAD` y `BODY`.

- Y dentro del `HEAD`...

- El elemento `TITLE`.

- Bien, puedes escribir las etiquetas en mayúsculas o en minúsculas, pero te recomiendo que las escribas siempre en minúsculas. Seguramente ya tendrás una estructura de este estilo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html40/strict.dtd">
<html>
 <head>
  <title></title>
 </head>
 <body>
 </body>
</html>
```

- Exacto, eso es lo que tengo.

- Pues es todo tuyo, ponle un título al documento en `TITLE` y crea tu contenido dentro de `BODY`. Utiliza los elementos que ya conoces. Asegúrate de que cierras los elementos en orden inverso al que los abres. Por ejemplo, `<p>Algo <em>enfatizado <strong>con fuerza</strong></em></p>`. Cuando acabes guarda el fichero en el disco duro. Lo puedes llamar por ejemplo "`ejemplo.html`" o como tú quieras. En todo caso acuérdate bien de dónde lo guardas.

- Vale, ya está guardado. ¿Ahora qué?

- Nada, abre el navegador que más te guste y abre el documento que acabas de guardar en el navegador. Y si lo has hecho todo bien, aparecerá tu documento como debería. Si no sale bien, repásalo. También puedes intentar validarlo para tratar de detectar errores, aunque lo más probable es que te líes más.

Nota: *si quieres, puedes enviarnos tu ejemplo por correo electrónico a html arroba conclase punto net o a través de nuestro formulario. Te ayudaremos encantados.*

- Bueno, yo creo que está bien, pero ya sabes, esto es un poco feo así...

- ¿Te gustaría que le pusiéramos unos estilos bonitos?

- Claro que sí. ¿Ahora mismo?

- Ahora mismo.

Creación de una hoja de estilo CSS

- Muy bien, vas a crear tu primera hoja de estilo.

- ¿Qué necesito?

- Nada. Crea un nuevo documento con el editor de textos que has usado para el documento HTML. La hoja de estilo también es un documento de texto sin formato.

- Ah, genial. Ya está.

- Ahora, vamos a especificar unas reglas de estilo para el cuerpo del documento, es decir, para el elemento `BODY`.

- Muy bien, ¿cómo?

- Veamos cierta terminología básica de CSS. Abre bien las orejas y concéntrate. Un *conjunto de reglas* tiene dos partes: *selector* y *bloque de declaraciones*. El selector especifica los elementos a los que se aplica el bloque de declaraciones. El bloque de declaraciones empieza con una llave abierta y termina con una llave cerrada. Entre las llaves hay cero o más *declaraciones* separadas por un punto y coma. Cada declaración es de la forma `propiedad : valor`. La especificación de CSS define las propiedades existentes y los valores que pueden tomar éstas.

- ¡Jo tío, qué lío me acabas de armar en un momento!

- Tranquilo, vamos a aclararlo con un ejemplo. Vamos a escribir un bloque de declaraciones para el `BODY`. El selector será el nombre del elemento:

```
body { }
```

Repasa las definiciones que te acabo de dar. ¿Está todo bien?

- Sí, ya, pero eso no es muy útil, ¿no?

- No, tenemos que poner unas declaraciones entre las llaves. Para no complicarnos mucho, podemos cambiar los colores y el tipo de letra, ¿te parece?

- Me parece.

- Bien. Para cambiar el color del texto, utilizamos la propiedad `color`. Para cambiar el color del fondo utilizamos la propiedad `background-color`, y para cambiar el tipo de de letra utilizamos la propiedad `font-family`.

- Jopé, siempre todo en inglés...

- ¡Qué le vamos a hacer! Escribe esto con el editor:

```
body {
 color : rgb();
 background-color : rgb();
 font-family : Verdana, Arial, Helvetica, Helv, sans-serif;
}
```

`rgb()` sirve para especificar un color a partir de sus componentes rojo-verde-azul. Si no sabes cómo funciona esto, te recomiendo que visites por ejemplo esta página de la Guía de la Computación, o más a fondo en la página de Juan Martínez-Val. ¿Qué colores te apetecen?

- No sé, algo que no sea demasiado hortera. Letras rojo oscuro sobre fondo naranja, por ejemplo.

- Entonces pon esto:

```
body {
 color : rgb(51,0,0);
 background-color : rgb(255,204,102);
 font-family : Verdana, Arial, Helvetica, Helv, sans-serif;
}
```

Ahora grábalo con el nombre "`estilo.css`" en el mismo directorio que el documento HTML.

- Ok, ya está. ¿Ahora que hay que hacer?

- Ahora hay que vincular esta hoja de estilo con nuestro documento HTML. Para ello, vamos a utilizar un nuevo elemento, el

elemento LINK, que debe estar contenido en el elemento HEAD. Escribe esto dentro del HEAD:

```
<link rel="stylesheet" href="estilo.css" media="screen" type="text/css" title="Mi hoja de estilo">
```

Observa que este elemento no tiene etiqueta final. No es que sea opcional o se me haya olvidado. Es que hay algunos elementos que no pueden tener etiqueta final.

- Vaya hombre. ¿Y qué significa todo eso exactamente?

- Significa que queremos vincular (link) la hoja de estilo (stylesheet) contenida en el fichero llamado "estilo.css" que está escrita en el lenguaje CSS (text/css) que sólo vale para pantallas de ordenador (screen) y que se titula (title) "Mi hoja de estilo". Esto del título tiene su importancia, pero ya te lo explicaré más adelante.

- Ok, perfecto. Ya lo he añadido. ¿Lo guardo?

- Sí, ya está. Con eso bastará. Las propiedades se heredarán desde el elemento padre BODY hasta sus elementos hijos, que son los que están contenidos en BODY: los párrafos, las listas, etc. Guárdalo y vuelve a abrir la página con el explorador. ¿Qué tal?

- Impresionante. Espera, me están empezando a llorar los ojos...

- ¡Anda, exagerado!

- No, en serio, tienes razón, esto es muy sencillo. El problema es que yo no sé qué elementos existen, ni que propiedades de estilo, ni nada...

- Pero por eso estás siguiendo el tutorial, ¿no? Como te dije al principio, prefiero que construyas tus conocimientos sobre bases sólidas, aunque avancemos mucho más despacio. En realidad, podría haberte explicado HTML en dos horas, pero seguro que habrías acabado usando todos esos truquitos miserables de los que te he hablado antes. Ten paciencia, ¿vale? ¡Lo hago por tu bien!

- ¡Pero qué padrazo estás hecho! Bueno, supongo que tienes razón.

- Creo que por hoy ya es más que suficiente. La próxima vez hablaremos de identificadores e hipervínculos, y haremos una especie de portal sencillito estilo Yahoo, así que prepárate una lista con tus páginas favoritas.

- ¡Eh, muy bien!

- ¡Hasta la vista!

- ¡Adiós!

Identificadores e Hipervínculos (I)

Introducción. Recursos y URIs

- Hola, ya estoy aquí otra vez.

- Hola, yo también. ¿Qué tal?

- Bien, gracias... ¿Preparado para el siguiente tutorial?

- Espero que sí. Hoy toca hablar de identificadores e hipervínculos...

- Sí, ¿has preparado una lista de tus páginas favoritas como te dije?

- Sí, está todo listo. ¿Empezamos?

- Muy bien. Si recuerdas, una de las primeras cosas que te dije en el primer tutorial es que cada recurso disponible en la Web necesita un identificador único que le diferencie de todos los demás.

- Sí, ya me acuerdo.

- Bien, ese identificador se llama *Identificador Uniforme del Recurso*, en inglés *Uniform Resource Identifier*, y sus siglas, por las que se le conoce habitualmente son URI.

- O sea, que cada recurso tiene un URI, ¿no?

- Exacto.

- Y cuando dices "recurso", ¿a qué te refieres exactamente?

- A cualquier cosa. Puede ser una página web, una imagen, un sonido, un vídeo, un fichero almacenado en una computadora, un mensaje de un grupo de noticias, una dirección de correo electrónico, un párrafo de un página de un libro electrónico... Pero

no sólo eso: también puede ser algo que no esté en la red: un libro, una organización, incluso una persona.

- Vaya... je, qué cosas...

- Sí, es lo que yo digo. El caso es que dar con una forma de poner nombres a cosas tan diferentes entre sí no es tan sencillo, y por eso tuvieron que inventar un sistema completo con sus reglas. Y me gustaría explicarte un poco las reglas para que las conozcas.

- Muy bien, pues ¡adelante con ello!

URNs y URLs. Esquemas de URLs

- Bien. Existen dos tipos de URI: los URN y los URL.

- Empezamos bien. ¿En qué se distinguen?

- Existen dos maneras distintas de identificar un recurso, según la finalidad que se persiga: podemos identificar un recurso por su nombre, o podemos identificarlo por su localización. El URN es un nombre ("N" de "name", "nombre") y el URL es un localizador ("L" de "locator", "localizador").

- Bien, veamos, a ver si lo entiendo. Si yo soy un recurso, mi URN sería mi nombre propio, y mi URL sería por ejemplo mi domicilio, ¿no?

- Sí, bueno, al menos básicamente esa es la idea. Nosotros en nuestras páginas web sólo vamos a usar URLs, así que de los URNs nos olvidamos.

- Muy bien, olvidados.

- Los URLs tienen una sintaxis que depende del tipo de recurso.

- Explícate un poco, anda.

- A ver. Por ejemplo, los URLs de las páginas web tienen una sintaxis que no es igual que la de los URLs de las direcciones de

correo electrónico. Según el tipo de recurso, el URL se estructura según un esquema diferente.

- Creo que te entiendo, pero me estoy perdiendo. Mejor que me pongas algún ejemplo.

- Eso está hecho. Esto es un URL de una página web:

```
http://www.conclase.net/index.html
```

Y esto es un URL de una dirección de correo electrónico:

```
mailto:html@conclase.net
```

- Ya veo. Entonces lo que pones a la izquierda te dice el tipo de recurso...

- ...y según el tipo de recurso, la parte de la derecha tiene una sintaxis diferente.

- Mmm, sí, entiendo. Es fácil.

- Entonces vamos a profundizar un poco más...

URLs genéricos y opacos. El esquema http

- Si te has fijado, los URLs del tipo `http` están formados por un grupo de palabras separadas por barras inclinadas, como por ejemplo, `http://html.conclase.net/tutorial/index`. A estos URLs se les llama *URLs genéricos*, y a los que no son así, se les llama *URLs opacos*. Por ejemplo, los URLs de tipo `mailto` son URLs opacos.

- Er... ¿y eso me lo dices o me lo cuentas?

- Verás, el motivo por el que se distingue entre unos y otros es porque los URLs genéricos pueden ser relativos...

- Te advierto que no me entero de nada de lo que dices...

- Muy bien, te voy a poner un ejemplo.

- Sí, sí, un ejemplo.

- Está bien, vamos a hablar de los URLs del tipo `http`, que son los que se refieren a páginas web y al resto de los documentos que se transmiten por el protocolo HTTP. Esos son los que usarás con más frecuencia en tus páginas web.

- Ok, soy todos oídos.

- La sintaxis de los URLs `http` sigue el siguiente esquema:

```
http://conclase.net:80/html/recursos/pagina.html?parametro
```

Este URL tiene cinco partes diferentes:

1. `http` es como sabes el nombre del esquema, dice qué tipo de URL le sigue.
2. `conclase.net` es el nombre de la computadora en que está almacenado el recurso. En realidad no tiene por qué ser un nombre de dominio, también puede ser un número, pero de eso no nos preocupamos.
3. `:80` es el número del puerto de la computadora por el que salen los documentos transmitidos por el protocolo `HTTP`. 80 es el valor por defecto, si no pones nada se supone que es el 80.
4. `/html/recursos/pagina.html` es la ruta de acceso al recurso en cuestión. Esto es parecido a los ficheros de tu disco duro. En este caso es algo así como el fichero `pagina.html` que está en el directorio (o carpeta) `recursos` que a su vez está en el directorio `html` que a su vez está contenida en el directorio principal o *raíz* del dominio.
5. Por último, `parametro` es una cadena de parámetros que recibe algún programa que esté en el servidor. De momento pasamos de eso, ya hablaremos en el futuro.

¿Está todo claro hasta aquí?

- Sí, creo que sí.

- Bien. Este URL era un *URL absoluto*. Te dice en qué máquina está el documento y en que directorio. Pero muchas veces es más práctico utilizar URLs relativos...

- Pero a ver, ¿qué es un URL relativo?

- Un *URL relativo* te da la localización de un recurso con respecto a la localización del recurso que contiene al URL.

- ¿Mmmm...?

- Para ponerte un ejemplo gráfico, imagínate esta situación: estamos en la cocina de tu casa y te pregunto: "¿Dónde está el ketchup?"; y tú me contestas: "en la nevera, en el segundo estante". Pero también me podrías haber dicho "En el Universo, en la Vía Láctea, en el Sistema Solar, en el Planeta Tierra, en el continente Europa, en el país España, en la ciudad Madrid, en la calle José Abascal, en el portal número 140, en el piso 4, en la puerta C, en la cocina, en la nevera, en el segundo estante".

- Je je, no, no creo que te hubiera dicho todo eso, pero creo que te entiendo. Lo primero sería un URL relativo, y lo segundo un URL absoluto, ¿no?

- Exacto. Lo primero, "en la nevera, en el segundo estante", sería un URL relativo, con relación a la cocina. Como veo que lo has entendido, vamos a ver más a fondo la sintaxis para los URLs relativos.

- Muy bien.

URLs absolutos y relativos

- Hablemos de URLs relativos. Hemos dicho que los URLs relativos sólo se pueden usar con URLs genéricos, como los del tipo `http`.

- Sí, lo dijiste antes, aunque no me enteré muy bien.

- Todos los URLs genéricos tiene una *ruta de acceso*. Por ejemplo, en el URL `http://conclase.net/html/recursos/pagina.html`, ¿cuál sería la ruta de acceso?

- Pues, según lo que me has dicho antes, sería lo que va entre el número del puerto y los parámetros. Supongo que en este caso sería `/html/recursos/pagina.html`.

- Sí señor. Pues bien, el *URL base* de un recurso es todo lo que hay en el URL absoluto del recurso, hasta la última barra inclinada inclusive. O sea, que en este ejemplo, el URL Base sería `http://www.conclase.net/html/recursos/`

- Ya... ¿y si no hubiera ninguna barra, por ejemplo, `http://www.conclase.net`?

- No, siempre hay una barra. En realidad es `http://www.conclase.net/`. Este URL se refiere al fichero que se carga por defecto en el directorio raíz (el directorio `/`) del dominio.

- Entiendo. En este caso el URL base es igual que el URL absoluto, ¿no?

- Sí. Bueno, pues el URL base es la localización con relación a la cual se declaran los URLs relativos. Como la cocina del ejemplo de antes.

- Mmm, necesito un ejemplo.

- Sí, imagina el recurso `http://www.conclase.net/docs/html/doc1`. Su URL base es `http://www.conclase.net/docs/html/`. Entonces si escribes en `doc1` el URL relativo `doc2`, estás haciendo referencia al documento `http://www.conclase.net/docs/html/doc2`.

- Ya veo. Añades el URL relativo al URL base y te da el URL absoluto, ¿no?

- Exactamente. Por ejemplo, el URL `manual/css/` correspondería al directorio `http://www.conclase.net/docs/html/manual/css/`, etc. Como ves los directorios siempre acaban con una barra, y los ficheros sin una barra.

- Entiendo.

- De todos modos, no siempre es tan directo como añadir el URL relativo al URL base, hay algunas excepciones...

- ¡Cómo no!

- El directorio ".." se refiere al directorio padre. En nuestro ejemplo, `../css/` correspondería a `http://www.conclase.net/docs/css/`.

- Ya, es como si quitaras del URL base todo lo que hay después de la penúltima barra.

- Sí, justo. Puedes concatenar varios hasta llegar al raíz, por ejemplo: `../../index.html` corresponde a `http://www.conclase.net/index.html`

- Muy bien, está claro.

- El directorio "." se refiere al directorio actual, o sea, que es lo mismo `./hola.txt` que `hola.txt`

- Muy bien, no le veo mucha lógica, pero bueno.

- Un URL relativo que comienza con "`/`" sustituye a toda la ruta de acceso del URL base. Te quedas sólo con el nombre de la máquina y el puerto.

- A ver, a ver, entonces `/hola.txt` sería `http://www.conclase.net/hola.txt`, ¿no?

- Sí. Y por último, un URL que comienza con "`//`" sustituye a todo lo que hay en el URL base desde el nombre de destino incluido.

- Por ejemplo, `//www.yahoo.com/` correspondería a `http://www.yahoo.com/`, ¿no?

- Sí, muy bien. Pues ya está. Eso es todo lo que tienes que saber sobre URLs relativos.

- Creo que lo entiendo más o menos bien.

- Como ves usando URLs relativos nos podemos ahorrar teclear un montón, pero sobre todo nos permite aislar partes enteras de un sitio web de su contexto. Por ejemplo, trabajando con URLs relativos, puedes crear tu sitio web en el disco duro de tu ordenador, puedes probar todos los vínculos y navegar perfectamente de unas páginas a otras, porque sólo usas caminos relativos: retrocede dos directorios, entra en este directorio, lee este fichero. Da igual que las páginas estén en tu ordenador, en el mío, en `www.conclase.net` o en `www.rediris.es`

- Pues tienes razón, no se me había ocurrido...

- Bueno, por ahora tenemos URLs para rato. Vamos a ponerlos en práctica...

Hipervínculos en HTML

- En el tutorial anterior te dije que para crear hipervínculos en HTML se utiliza el elemento `A`.

- Sí, lo recuerdo.

- Bien, entonces es muy fácil. Un hipervínculo tiene tres características que lo definen: de dónde sale, hacia dónde va, y el tipo de vínculo.

- Las dos primeras las comprendo, pero la tercera no.

- En realidad la tercera característica, el tipo de vínculo, no es excesivamente importante. Por ejemplo, en el caso típico de un vínculo que apunta al capítulo siguiente, el tipo de vínculo es "Next" ("Siguiente"). Pero no nos preocupemos por eso de momento.

- Muy bien, lo que tú digas.

- El elemento de HTML `A` permite determinar estas características. ¿De dónde sale? De donde hayamos puesto el elemento en el código. ¿A dónde va? A donde diga el valor del

atributo `href` del elemento. Incluso te permite determinar el tipo de vínculo.

- Ya veo. Y el valor del atributo `href` es justamente un URL, ¿no?

- No exactamente. Concretamente, es una *referencia URL*. Una referencia URL es un URL al que opcionalmente se le puede añadir un *identificador de fragmento*. Por ejemplo, imagina que quieres hacer un hipervínculo al cuarto párrafo de una página web. Pues sólo tienes que dar un identificador de fragmento a ese párrafo, y para apuntar a él añades al URL de esa página web el identificador de fragmento, separado por el símbolo `#`.

- Ya... ¿por ejemplo?

- Por ejemplo, `<a href="../manual/instalacion.html#montaje">`

- Ya veo. ¿Y cómo se asigna un identificador de fragmento a un... fragmento?

- Pues, ¡con el elemento `A`!

- ¡Ah, claro, ya me acuerdo! Lo vimos en el tutorial anterior. Para que un elemento `A` sea el origen de un hipervínculo, hay que darle un valor a su atributo `href`, y para que sea el destino de un hipervínculo, hay que darle un valor a su atributo `name`.

- ¡Sí, muy bien! En realidad también puedes usar el atributo `id` en lugar del atributo `name`. Por ejemplo:

```
<p><a id="montaje">El montaje</a> del equipo es sencillo y no necesita herramientas especiales, etc., etc.</p>
```

- Sí, ya entiendo.

- Por cierto, si el autor de una página no ha definido destinos de vínculo en su página, difícilmente podrás crear hipervínculos a partes de esa página. Y lo malo es que en general casi nadie define destinos a menos que le hagan falta a él, lo cual es una pena, porque así desperdiciamos la capacidad de la Web. Así que, si eres

tan amable, cuando ya sepas HTML y diseñes páginas de verdad, no te olvides de poner identificadores de fragmento en los sitios que puedan resultar interesantes, ¿vale?

- Venga, vale, lo haré por ti.

- No, por mí no, por la Web, y por ti mismo. Así será más fácil hacer enlaces a tus páginas y tus páginas serán más fáciles de usar y por tanto más útiles.

- Vale, vale, si ya me habías convencido...

- Venga, no nos enrollemos. Prepara tus enlaces y abre tu editor de textos. ¡Vamos a hacer nuestro pequeño primer portal!

- ¡Bien, bien!

Confección de un miniportal de ejemplo

- Tengo la impresión de que tu portal va a tratar de música celta...

- Sí, ¿cómo lo has sabido?

- No, por nada... Bueno, ya sabes cómo empezar, ¿no?

- Sí, a ver si lo encuentro... Así:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html40/strict.dtd">
<html>
 <head>
  <title>El portalillo de música celta</title>
 </head>
 <body>
 </body>
</html>
```

- Muy bien. Vamos a empezar a rellenar el cuerpo. Cómo ya sabes, nos olvidamos de momento de la apariencia y pensamos sólo en términos de estructura lógica de los contenidos.

- Bien, entonces lo primero que voy a poner es un encabezado `H1`. Me gusta este nombre, es de una de mis canciones favoritas:

```
<body>
 <h1>Port An Deoraí</h1>
</body>
```

- Buena idea.

- Me gustaría ponerle un subtítulo. ¿Qué elemento debería usar?

- Mmm, buena pregunta. Podrías usar un elemento de encabezado, pero yo usaría un elemento `P`.

- Sí, es lo que yo había pensado.

```
<body>
 <h1>Port An Deoraí</h1>
 <p>El portalillo de música celta</p>
</body>
```

- Muy bien. ¿Cómo has organizado los enlaces?

- Por zonas geográficas.

- Bien, pues escribe las zonas que tengas pensado. De momento no pongas ningún vínculo.

- Vale:

```
<body>
 <h1>Port An Deoraí</h1>
 <p>El portalillo de música celta</p>

 <h2>Irlanda</h2>
 <h2>Escocia</h2>
 <h2>Bretaña</h2>
 <h2>España</h2>
 <h2>Estados Unidos y Canadá</h2>
</body>
```

- Lo ideal sería tener una página para cada zona, en la que pusieras una lista con los grupos cada uno enlazado con su página web. En la portada puedes poner debajo de cada zona tres o cuatro grupos.

- A ver, voy a poner los típicos, así:

```
 <body>
  <h1>Port An Deoraí</h1>
  <p>El portalillo de música celta</p>

  <h2>Irlanda</h2>
   <p>Lúnasa, Clannad, Altan, Dervish...</p>
  <h2>Escocia</h2>
   <p>Andy M. Stewart, Battlefield Band, Tannahill
Weavers...</p>
  <h2>Bretaña</h2>
   <p>Gwendal, Strobinell, Diwall...</p>
  <h2>España</h2>
   <p>Milladoiro, Na Lúa, Luar Nalubre...</p>
  <h2>Norteamérica</h2>
   <p>La Bottine Souriante, Fine Crowd, Tempest...</p>
 </body>
```

- Muy bien. Yo creo que es el momento de poner los vínculos; orígenes de vínculo para ser más concretos. Por ejemplo, `<a href="el/URL/de/lunasa">Lúnasa</a>`, y así todos. Ten cuidado con las mayúsculas y las minúsculas: en el nombre de la máquina (el dominio) no se distingue, pero en la ruta de acceso sí. No es lo mismo `/docs/doc1.doc` que `/DOCS/doc1.doc`.

- Vale, lo voy a ir haciendo.

- A ver qué tal queda...

Otros esquemas de URLs

- ¿Ya has acabado?

- Sí aquí tienes el código completo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
```

```
"http://www.w3.org/TR/html40/strict.dtd">
<html>
 <head>
  <title>El portalillo de música celta</title>
   <link rel="StyleSheet" href="ejemplo5.css" media="screen"
type="text/css">
 </head>
 <body>
  <h1>Port An Deoraí</h1>
  <p><em><strong>El portalillo de música celta</strong></em>
</p>

        <h2><a    href="http://www.ceolas.org/artists/index-
ie.html">Irlanda</a></h2>
   <p><a href="http://www.lunasa.ie/">Lúnasa</a>,
                                                          <a
href="http://www.ceolas.org/artists/Clannad/">Clannad</a>,
      <a href="http://www.altan.ie/">Altan</a>,
      <a href="http://www.dervish.ie/">Dervish</a>...</p>

        <h2><a    href="http://www.ceolas.org/artists/index-
sc.html">Escocia</a></h2>
        <p><a   href="http://www.andymstewart.com/">Andy   M.
Stewart</a>,
                                                          <a
href="http://www.battlefieldband.co.uk/">Battlefield
Band</a>,
        <a href="http://www.tannahillweavers.com/">Tannahill
Weavers</a>...</p>

   <h2><a  href="http://www.gwerz.com/index.html">Bretaña</a>
</h2>
                                                        <p><a
href="http://www.gwerz.com/artistes/biographies/gwendal.htm"
```

```
>Gwendal</a>,
                                                                    <a
href="http://www.arbedkeltiek.com/galleg/musique/strobinell.
htm">Strobinell</a>,
                                                                    <a
href="http://www.bmol.infini.fr/adherent/diwall/index.fr.htm
">Diwall</a>...</p>


                                                              <h2><a
href="http://www.arrakis.es/~josugp/folk.htm">España</a>
</h2>
   <p><a href="http://www.milladoiro.com/">Milladoiro</a>,
      <a href="http://www.nalua.net/">Na Lúa</a>,
                <a   href="http://www.luarnalubre.com/">Luar
Nalubre</a>...</p>

        <h2><a    href="http://www.ceolas.org/artists/index-
us.html">Estados Unidos</a> y
              <a  href="http://www.ceolas.org/artists/index-
ca.html">Canadá</a></h2>
                                                               <p><a
href="http://www.millepattes.com/Anglais/bottine/">La
Bottine Souriante</a>,
         <a  href="http://celtic.relics.com/finecrowd/">Fine
Crowd</a>,
       <a href="http://www.tempestmusic.com/">Tempest</a>...
</p>
 </body>
</html>
```

- Vamos a ver cómo queda.
- Mmm, vaya, ya sabes lo que opino yo de esto...

- Sí, ya lo sé. Pero no te preocupes, había preparado una hoja de estilo para que no te desanimaras. Es una cosa sencillita, no esperes grandes cambios: aquí lo puedes ver.

- Bueno, queda resultón.

- Se podrían hacer cosas más complicadas, como poner gráficos y tal, pero por ahora no está mal para ser el tercer tutorial. Bueno, antes de acabar estaría bien que habláramos un poco de los otros tipos o esquemas de URLs además del `http`.

- Bueno, pero no te enrolles mucho que esto ya se está haciendo un poco largo.

- Muy bien. Hemos hablado ya del esquema `mailto`, que se refiere a buzones de correo electrónico. Por ejemplo, podrías crear un hipervínculo a un buzón de correo: `<a href="mailto:lopez@example.com">La dirección de López</a>`

- Vaya, ¿y podría entrar en el buzón y leer tus mensajes si siguiera ese hipervínculo?

- No hombre, lo normal es que se abra el programa de correo electrónico para enviar un mensaje a esa dirección de correo.

- Ah, claro, qué cenutrio soy.

- Seguimos. El esquema `ftp` es muy parecido al `http`, y se usa para ficheros que se pueden descargar por FTP.

- ¿Qué es FTP?

- FTP son las siglas de "File Transfer Protocol" ¡Sí, ya lo sé, está en inglés! "Protocolo de transferencia de ficheros", si te quedas más tranquilo...

- Vale, vale...

- El protocolo HTTP sirve para transportar páginas de hipertexto, el protocolo FTP sirve para transportar ficheros en general.

- Je, hay que reconocer que poniendo nombres a las cosas son unos hachas...

- Un URL del tipo `ftp` podría ser por ejemplo éste:

```
ftp://ftp.funet.fi/pub/standards/RFC/rfc2396.txt
```

Eso es un fichero de texto que está disponible a través de un servidor FTP. Además dentro de un URL `ftp` también puedes incluir el usuario y su contraseña, así:

```
ftp://usuario@ftp.servidor.es/pub/loquesea.zip
ftp://usuario:clave@ftp.servidor.es/pub/loquesea.zip
```

- Ya veo, no me entero muy bien, pero bueno.

- Pronto te enseñaré a usar el FTP, es la manera más normal de subir tus páginas a tu sitio web. Bueno, ya queda poco. El esquema `file` se usa para ficheros almacenados en un ordenador. De momento para nosotros no tiene mucho interés, pero también es similar a los URLs `http`. Por ejemplo, si estás en Windows, este URL se refiere al fichero `c:templeeme.txt` (Windows usa barras invertidas):

```
file://localhost/c:templeeme.txt
```

Como ves es muy parecido, primero el nombre de la máquina y luego la ruta de acceso.

- Ya. Qué cosas más raras.

- Más. El esquema `news` se refiere a grupos de noticias o a mensajes de grupos de noticias. Según se refiera a una cosa o a la otra, la sintaxis varía. Por ejemplo, para un grupo de noticias:

```
news:es.comp.infosistemas.www.paginas-web
```

y para un mensaje en particular:

```
news:3b3c46bd.3498640@news.cis.dfn.de
```

- Sí, conozco ese grupo. Por cierto, estas URL son opacas, ¿no?

- Jo, a veces me dejas alucinado. Sí son URLs opacas. Bueno, y para acabar, el esquema `telnet`. Telnet es un protocolo que te permite acceder a una computadora a través de la red. Un URL

`telnet` se refiere a una sesión telnet. La sintaxis es una especie de mezcla entre la de `http` y la de `ftp`:

```
telnet://usuario:clave@servidor.es:35/
```

- Vale, aunque si te soy sincero lo del telnet es la primera vez que lo veo en mi vida.

- Quién sabe, quizás en el futuro uses telnet con más frecuencia de lo que te imaginas. Bueno, creo que ya hemos acabado.

- Uf, por fin...

- Pues había pensado hacer en el próximo tutorial un repaso de los elementos de HTML, que ya va siendo hora, así que prepárate para otra tutorial largo...

- Bueno, pero espero que sea menos abstracto que este. Estoy de URLs hasta el moño, la verdad.

- ¡Qué exagerado eres!

- Y tú que lo digas. Bueno, hasta la próxima.

- ¡Hasta luego!

Elementos de HTML

Introducción

- Hola.

- Hola.

- Hoy estarás contento, por fin vamos a hablar de los tipos de elemento de HTML.

- Sí, la verdad es que ya tenía ganas.

- Lo sé. Después del tutorial de hoy ya podrás hacer lo que quieras.

- Je je, no creo, pero bueno.

- Bueno, por lo menos tendrás un conocimiento general y estarás en posición de leer la especificación sin perderte demasiado. Pero vamos a empezar. Podemos agrupar los elementos de HTML en los siguientes grupos:

- Estructura
- Texto
- Listas
- Tablas
- Vínculos
- Objetos
- Estilo
- Marcos
- Formularios
- Scripts

Vamos a hablar un poco de cada grupo, así que ya te puedes suponer que el tutorial de hoy será un poco largo. Tómate tu tiempo.

- Bueno, todo sea por amor al conocimiento.

- Una cosa antes de empezar: vamos a hacer un repaso de la mayor parte de los elementos, pero no de sus atributos, porque si no esto se convertiría en la especificación, y no es ése el plan. Si quieres saber cuáles son los atributos que puede tener un elemento, te he puesto un enlace entre cada elemento y su definición en la traducción de la especificación.

- Ah, muy bien.

- Además en la especificación tienes una lista con todos los elementos y una lista con todos los atributos. Bueno, pues si estás preparado, vamos allá.

- Allá vamos.

Elementos de estructura

- Empezamos con los *elementos de estructura*. Estos ya los conoces y son `HTML`, `HEAD` y `BODY`.

- Sí, je je, de algo me suenan...

- El elemento `HTML` es el elemento raíz del documento, que representa al documento completo:

Elemento	`HTML`
Modelo de contenido	Un elemento `HEAD` y un elemento `BODY`, en ese orden
Etiqueta inicial	opcional
Etiqueta final	opcional

- ¿Qué es eso de modelo de contenido?

- El *modelo de contenido* es lo que puede aparecer como contenido del elemento, es decir, entre las etiquetas inicial y final. ¿Recuerdas que hablamos de elementos en bloque y en línea?

- Sí.

- Te dije que esa clasificación era útil para describir lo que podía contener cada tipo de elemento. Lo verás a lo largo de este tutorial. Al final, cuando hayamos visto todos los elementos, te diré a qué grupo pertenece cada uno. Hasta entonces considera que los elementos en línea forman parte de una línea (como por ejemplo una palabra enfatizada) y los elementos en bloque tienen un salto de línea antes y otro después (como por ejemplo un párrafo o una lista). Las letras y palabras sueltas se consideran elementos en línea.

- Muy bien.

- Seguimos. El elemento `HEAD` contiene la información de cabecera del documento:

Elemento	`HEAD`
Modelo de contenido	cero o más elementos `SCRIPT`, `STYLE`, `META`, `LINK` u `OBJECT`; además obligatoriamente un elemento `TITLE`, y opcionalmente un elemento `BASE`
Etiqueta inicial	opcional
Etiqueta final	opcional

Como ves el elemento `HEAD` es como un cajón de sastre donde metemos muchos tipos de información.

- Sí, ya me estoy fijando.

- De momento ahora hablamos de `TITLE` y `META`. `TITLE` es el título del documento como ya sabes. Debes intentar que sea lo más descriptivo posible, pero sin pasarte:

Elemento	TITLE
Modelo de contenido	cualquier clase de caracteres, pero no código
Etiqueta inicial	opcional
Etiqueta final	opcional

Es decir, puedes poner letras normales, letras con acentos, etc. Y símbolos. Por ejemplo, el símbolo de copyright. Lo único malo es que como los teclados no suelen tener el símbolo de copyright, se usa en su lugar un código especial: `&copy;`. Hay muchos más códigos especiales que empiezan por `&` y terminan por `;`. Se llaman *referencias a entidades de caracteres* y hablaremos de ellas en el futuro.

- Vale.

- En cuanto al elemento META, me temo que es un poco más complicado. Se usa sobre todo para poner una descripción y una lista de palabras clave en tu página web, pero tiene otros usos, aunque la mayoría no son recomendables.

Elemento	META
Modelo de contenido	VACIO

Como ves es un elemento vacío, no puede tener etiqueta final.

- Entonces, ¿dónde va la información?

- En atributos, por ejemplo:

```
<meta name="keywords" contents="tutorial,HTML,elementos">
```

Hay un tutorial sobre elementos META en HTML con Clase por si lo quieres ver, pero creo que la mayoría de las cosas de momento no te interesan. Si no entiendes nada no te desanimes.

- Bueno, le echaré un vistazo de todos modos.

- Vamos con el elemento BODY, el cuerpo del documento:

Elemento	BODY
Modelo de contenido	uno o más elementos en bloque y/o elementos SCRIPT; también pueden aparecer elementos INS y DEL
Etiqueta inicial	opcional
Etiqueta final	opcional

Eso es todo. Existen otros dos elementos de estructura, DIV y SPAN, de los que hablaremos más adelante en este tutorial.

- Muy bien, pues seguimos.

Frases y párrafos

- El siguiente grupo de elementos del que vamos a hablar se refiere principalmente a *frases* y *párrafos*. Comenzamos con los elementos de frase:

Elementos	EM, STRONG, DFN, CODE, SAMP, KBD, VAR, CITE, ABBR, ACRONYM
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

- Muy bien. Alguno me suena.

- Sí, alguno ya lo hemos hemos utilizado antes. Te describo su utilidad rápidamente:

`EM`:
: Enfatiza el contenido (*emphasis*). Normalmente se representa poniéndolo en cursiva, aunque, como todo en HTML, depende del navegador, y se puede modificar con hojas de estilo.

`STRONG`:
: Indica un énfasis más fuerte. Normalmente se representa poniendo el contenido en negrita.

`CITE`:
: Contiene una cita o una referencia a otras fuentes.

`DFN`:
: Indica que aquí es donde se define el término encerrado.

`CODE`:
: Designa un fragmento de código de computadora.

`SAMP`:
: Designa una muestra de la salida de un programa, script, etc.

`KBD`:
: Indica texto que debe ser introducido por el usuario.

`VAR`:
: Indica que el texto es una variable o un argumento de un programa.

`ABBR`:
: Indica que el contenido es una abreviatura (p.ej., WWW, HTTP, URL, Sr., etc.). Esto puede ser útil porque en el atributo `title` del elemento se puede poner el significado de la abreviatura.

`ACRONYM`:
: Indica un acrónimo (p.ej., RENFE, radar, etc.).

En la práctica, los que más utilizarás son `EM` y `STRONG`, y con menos frecuencia `CITE`, `DFN` y `ABBR`. ¿Vamos bien?

- Más o menos.

- Bien. Seguimos con los elementos de párrafo. Básicamente son dos:

Elemento	P
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	opcional

- Sí, esto ya estamos hartos de usarlo. ¿Cuál es el otro?

- El otro es el elemento PRE, para *texto preformateado*.

- ¿Ein? ¿qué es eso?

- Como ya sabes, todo lo que escribas entre y se representa como un párrafo. Aunque pongas saltos de línea en el código HTML, al representarlo saldrá un sólo párrafo. Aunque pongas diez espacios seguidos, al representarlo saldrá uno solo.

- Sí, eso ya lo sé.

- Con el elemento PRE, los saltos de línea y los espacios en blanco que dejes saldrán representados. Esto es útil por ejemplo para escribir código de programas.

Elemento	PRE
Modelo de contenido	cero o más elementos en línea, excepto IMG, OBJECT, BIG, SMALL, SUB, SUP
Etiqueta inicial	obligatoria
Etiqueta final	obigatoria

- Ya veo.

- En realidad hay una manera de provocar un salto de línea en un punto predeterminado usando el elemento P, que es con el

elemento `BR`:

Elemento	`BR`
Modelo de contenido	VACIO

Como es un elemento vacío, no puedes poner etiqueta final. Simplemente donde escribas `<br>` habrá un salto de línea.

- Pero tú me dijiste que un salto de línea es donde termina un párrafo y empieza otro.

- Sí. Recuerda que tratamos de separar la estructura de la apariencia. Por eso no te recomiendo que uses el elemento `BR`, a menos que sea absolutamente necesario.

- ¿Y si lo que queremos es impedir un salto de línea en un punto?

- Pues en vez de poner un espacio normal, pones un *espacio de no separación*.

- Ah, muy bien, ¿y cómo?

- Para escribir un espacio de no separación utilizamos el código ` ` (*non-breaking space*). Por ejemplo: `salto impedido`. Pero seguimos. Otros elementos que conocemos ya son los encabezados de sección:

Elementos	`H1`, `H2`, `H3`, `H4`, `H5`, `H6`
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	obigatoria

El encabezado `H1` es más importante que el `H2`, el `H2` más importante que el `H3`, y así sucesivamente hasta el `H6` que es el

menos importante..

- Sí, ya lo sé.

- Pues seguimos. Otro grupo de elementos de texto son las *citas*. Se distingue entre citas en línea (`Q`) y citas en bloque (`BLOCKQUOTE`):

Elemento	`Q`
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	obigatoria

Elemento	`BLOCKQUOTE`
Modelo de contenido	elementos en bloque, `SCRIPT`
Etiqueta inicial	obligatoria
Etiqueta final	obigatoria

El elemento `Q` es para citas cortas (frases, refranes...) y las citas largas contienen su propio párrafo o párrafos (como ves contiene elementos en bloque, no en línea, así que no puede contener palabras directamente).

- Ya. ¿Y qué diferencia hay entre `CITE` y `Q`?

- Bueno, lo normal es meter en `CITE` el origen de la cita (el autor, el libro, la persona que dijo eso, etc.) y en `Q` la cita en sí. De todos modos, tanto `Q` como `BLOCKQUOTE` tienen un atributo `cite` donde puedes poner el origen de la cita, aunque normalmente no aparecerá directamente representado.

-Bueno, vale.

- Por último, tenemos el elemento `ADDRESS`, que da información de contacto del autor, y cuando aparece suele hacerlo al principio del documento:

Elemento	`ADDRESS`
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	obigatoria

¿Seguimos?

- Seguimos.

Listas

- El siguiente grupo de elementos se refiere a *listas*. Ya las hemos usado antes...

- Sí, `OL` y `LI`, ¿no?

- Exacto, y hay algunos más. Existen tres tipos de lista: listas ordenadas, listas no ordenadas y listas de definiciones.

- ¿Qué es exactamente cada cosa?

- Para entender la diferencia entre listas ordenadas y no ordenadas el ejemplo más claro es el de la receta de cocina: los ingredientes los pondrías en una lista no ordenada, y los pasos para cocinar la receta en una lista ordenada, porque en ese caso el orden es importante.

- Claro, los ingredientes los puedes conseguir en cualquier orden, pero la receta la tienes que preparar en el orden correcto.

- Justo. Por eso normalmente en las listas ordenadas los objetos salen numerados y en las listas no ordenadas salen con un

marcador, como un circulito o un cuadradito.

Elementos	UL, OL
Modelo de contenido	uno o más elementos LI
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Los LI son como sabes los objetos de la lista (*list items*).

- Sí, pero oye, si dentro de una lista sólo puede haber elementos LI, ¿quiere eso decir que no se pueden anidar unas listas dentro de otras?

- No, mira:

Elemento	LI
Modelo de contenido	cero o más elementos en bloque y/o en línea
Etiqueta inicial	obligatoria
Etiqueta final	opcional

- Ah, o sea que si quiero anidar una lista tengo que meterla dentro de un elemento LI, ¿no?

- Exactamente. Bueno, las listas de definiciones (elemento DL, *definition list*) son ligeramente más complicadas. En estas listas aparecen dos tipos de objetos de lista: los términos definidos (DT, *definition term*) y las definiciones de los términos (DD, *definition description*). En el apartado anterior de este tutorial te he puesto una de estas listas con la utilidad de los elementos de frase.

Elemento	DL
Modelo	uno o más elementos

de contenido	`DT` o `DD`, en cualquier orden
Etiqueta inicial	obligatoria
Etiqueta final	opcional

Elemento	`DT`
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	opcional

Elemento	`DD`
Modelo de contenido	cero o más elementos en bloque y/o en línea
Etiqueta inicial	obligatoria
Etiqueta final	opcional

En la especificación de HTML tienes un ejemplo en que se utilizan al mismo tiempo los tres tipos de listas.

- Sí, está bastante bien.

- Bueno, pues ya hemos acabado con las listas.

- Qué rápido. Pues seguimos.

Tablas

- Hablamos ahora de las *tablas*. Las tablas se usan como su nombre indica para marcar datos tabulares. Las tablas pueden llegar

a ser bastante complejas, y de hecho tengo pensado dedicar un tutorial completo a hablar sólo de tablas. Pero lo normal es que las utilices para hacer tablas sencillas y entonces no tienen mucha dificultad.

- Bien, me alegro.

- El elemento que se usa es el elemento `TABLE`:

Elemento	`TABLE`
Modelo de contenido	los siguientes elementos en este orden: elemento `CAPTION` opcional, cero o más elementos `COL` y/o `COLGROUP`, un elemento `THEAD` opcional, un elemento `TFOOT` opcional, y uno o más elementos `TBODY`
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

- ¿Te importaría explicarme qué son todos esos elementos?

- Por supuesto que no. `CAPTION` es el título de la tabla, lo que dice qué es la tabla. Por ejemplo, "Evolución de la población durante los últimos 10 años".

Elemento	`CAPTION`
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta	obligatoria

final	

- Bien, eso está claro.

- COL y COLGROUP se usan para agrupar columnas, mientras que THEAD, TFOOT y TBODY se usan para agrupar filas. Lo mejor es que son todos opcionales, y para los usos normales lo más seguro es que no tengas que utilizarlos. Así que ya hablaremos de ellos en un futuro tutorial.

- Vale, pero no estoy muy de acuerdo contigo. Según lo que has puesto arriba, debe haber uno o más elementos TBODY dentro del elemento TABLE.

- Eeh, sí, tienes toda la razón, pero mira:

Elemento	TBODY
Modelo de contenido	uno o más elementos TR
Etiqueta inicial	opcional si sólo hay un TBODY y ningún THEAD y TFOOT
Etiqueta final	opcional

Lo que quiere decir esto es que en las tablas sencillitas, no hace falta poner las etiquetas del elemento TBODY.

- Ya entiendo. ¿Qué es TR?

- TR es una fila de la tabla (*table row*). Dentro de cada fila de la tabla hay varias celdas:

Elemento	TR
Modelo de contenido	uno o más elementos TD y/o TH
Etiqueta	obligatoria

inicial	
Etiqueta final	opcional

Como ves hay dos tipos de celdas: de datos (`TD`, *table data*), y de encabezado (`TH`, *table header*):

Elementos	`TD`, `TH`
Modelo de contenido	cero o más elementos en línea y/o en bloque
Etiqueta inicial	obligatoria
Etiqueta final	opcional

Para que lo veas con un ejemplo, este es un caso típico de tabla HTML:

```
<table>
 <tr>
  <th>Elemento</th>
  <td><code>TR</code></td>
 </tr>
 <tr>
  <th>Modelo de contenido</th>
  <td>uno o más elementos <code>TD</code>
      y/o <code>TH</code></td>
 </tr>
 <tr>
  <th>Etiqueta inicial</th>
  <td>obligatoria</td>
 </tr>
 <tr>
  <th>Etiqueta final</th>
  <td>opcional</td>
 </tr>
<table>
```

- Menudo lío. Y eso que es sencilla.

- Sí. Y aunque éstos son todos los elementos que hay para tablas en HTML, encima hay un montón de atributos. Hablaremos a

fondo en el futuro, de momento vamos a pasar al siguiente grupo de elementos.

- Muy bien.

Vínculos

- El siguiente grupo son los *vínculos*.

- Ah, eso está chupao.

- Sí, más o menos. El elemento `A` ya lo conocemos perfectamente:

Elemento	`A`
Modelo de contenido	cero o más elementos en línea, excepto otros elementos `A`
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Ya sabes, el atributo `href` para convertirlo en origen de vínculo y el atributo `name` para convertirlo en destino.

- ¿Ya está? ¿Pasamos al siguiente grupo?

- No, no tan rápido. Hay más. El elemento `LINK`, que si recuerdas se puede colocar dentro del elemento `HEAD` también pertenece a este grupo.

- Bueno, sí, en cierto modo vincula unos documentos con otros, porque es lo que usábamos para vincular la hoja de estilo, ¿no?

- Sí:

Elemento	`LINK`
Modelo de	VACIO

Además de vincular hojas de estilo, puedes vincular otros documentos. Esto se usa para poner en la información de cabecera del documento cuáles son los documentos que están relacionados con éste. Por ejemplo:

```
<head>
  <title>Capítulo 2</title>
  <link rel="Index" href="../indice.html">
  <link rel="Next"  href="Capitulo3.html">
  <link rel="Prev"  href="Capitulo1.html">
</head>
```

En vez de (o además de) especificar el atributo `rel` puedes especificar el atributo `rev`, que expresa un *vínculo inverso*. Por ejemplo, si en el documento `Capitulo2.html` pones esto:

```
<head>
  <title>Capítulo 2</title>
  <link rel="Glossary" href="glosario.html">
</head>
```

en el glosario también podrías expresar esa misma relación, así:

```
<head>
  <title>Glosario</title>
  <link rev="Glossary" href="Capitulo2.html">
</head>
```

El primero *dice "`glosario.html`" es mi glosario*, y el segundo dice *yo soy el glosario de "`Capitulo2.html`"*.

- Mmm, ya veo. Un poco raro. ¿Y qué cosas puedo poner en `rel` y `rev`?

- Puedes ver los tipos de vínculos en la especificación. El atributo `rel` también lo puedes poner en el elemento `A`.

- Ah, entonces a eso te referías en el tutorial anterior cuando decías que un vínculo se caracterizaba por su origen su destino y su tipo, ¿no?

- Sí, eso mismo. De momento el elemento `LINK` no lo utilizan muchos navegadores, pero podría ser útil para crear barras de

navegación automáticas. Como lo más probable es que eso suceda en el futuro, puedes acostumbrarte a usarlos. Además hay motores de búsqueda que los utilizan para indexar tu sitio web y para eso son muy útiles.

- Muy bien.

- Por último, el elemento `BASE` dice cuál es el URI base al que se añaden los URLs relativos que haya en el documento. De esto hablamos de sobra en el tutorial anterior y espero que no tengas muchas dudas:

Elemento	`BASE`
Modelo de contenido	VACIO

El URI base se especifica como el valor del atributo `href` del elemento. Y debe ser un URI absoluto.

- Muy bien.

- Perfecto, pues vamos al siguiente grupo.

Objetos incluidos

- El siguiente grupo de elementos son los relacionados con *objetos incluidos*.

- ¿Qué quiere decir eso?

- *Objetos incluidos* son imágenes, aplicaciones, documentos externos, y también las imágenes con zonas sensibles.

- Vaya, pues es un grupo muy grande, ¿no?

- Sí. En teoría valdría con un solo elemento, el elemento `OBJECT` que vale para todo, pero de momento no está bien implementado en los navegadores existentes y todavía hace falta usar los elementos antiguos, que son unos cuantos.

- Pues qué fastidio, ¿no?

- Sí, desgraciadamente es lo normal así que vete acostumbrando. Veamos el elemento OBJECT:

Elemento	OBJECT
Modelo de contenido	cero o más elementos PARAM y/o elementos en línea o en bloque
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Si vas a la especificación, verás que tiene unos atributos que dan un poco de miedo.

- Pues sí, un poquillo...

- La mayoría los aprenderás cuando aprendas a hacer aplicaciones que puedas poner en páginas web, así que de momento no vamos a verlo más a fondo.

- Bueno, me parece buena idea.

- Para algo más simple como poner una imagen en teoría bastaría algo así:

```
<object data="logo.gif" type="image/gif">
<strong>HTML con Clase</strong>
</object>
```

Así, si por cualquier motivo el usuario no puede cargar o no puede ver la imagen, en su lugar lo que aparece es lo que hay dentro del OBJECT. Podría ser otro elemento OBJECT, pero en este caso es texto simple. El problema es que como te he dicho, esto no funciona muy bien en los navegadores actuales...

- Sí, ¿y entonces?

- Entonces hay que utilizar el elemento IMG:

Elemento	IMG

Modelo de contenido	VACIO

El ejemplo anterior se convertiría en esto:

```
<img src="logo.gif" alt="HTML con Clase">
```

- Bueno, es más sencillo, ¿no?

- Sí, pero claramente es menos flexible. Pero bueno, cuando uses el elemento `IMG` no olvides que el atributo `alt`, que contiene el texto alternativo a la imagen, es obligatorio, y sólo puede contener letras, no código. Si tu imagen es un adorno que no tiene texto alternativo especificas `alt=""`, aunque entonces debería estar en una hoja de estilo. También es interesante especificar los atributos `width` (anchura) y `height` (altura) para acelerar la carga de las páginas.

- Vale, vale, no te embales.

- Tranqui, no vamos a hablar más de esto de momento. A este grupo pertenecen además los elementos `APPLET` (que está desaprobado), el elemento `PARAM` (para pasar parámetros a las aplicaciones), y los elementos `MAP` y `AREA` que se utilizan para crear imágenes con "zonas sensibles" o mapas de imágenes. Si quieres más información mira en la especificación donde hay un capítulo entero dedicado a todo esto.

- Bueno, vale.

- En el futuro tendremos un tutorial sobre esto. Pero vamos al grupo siguiente.

- Vamos, vamos.

Estilo

- El grupo siguiente está relacionado con la especificación de reglas de *estilo*.

- ¿Te refieres a estilos como en CSS?

- Sí. Hasta ahora siempre hemos puesto las reglas de estilo en una hoja de estilo separada y la hemos vinculado al documento HTML a través de un elemento `LINK`.

- Sí...

- Pero hay otras dos maneras de declarar reglas de estilo: con un elemento `STYLE` o con un atributo `style`.

- ¡Vaya! No me habías dicho nada...

- No, y en realidad te recomiendo que sigas haciendo como hasta ahora, porque si la idea es separar la apariencia de la estructura, no tiene mucho sentido poner las reglas de estilo en el documento HTML.

- No, claro...

- Como mucho, para hacer pruebas, el elemento `STYLE` sí puede ser práctico:

Elemento	`STYLE`
Modelo de contenido	información de estilo (depende del lenguaje)
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Esto se pone dentro del elemento `HEAD`, y en un atributo tienes que especificar el lenguaje de la hoja de estilo, por ejemplo para una hoja CSS, sería así:

```
<style type="text/css">
<!--
 reglas de estilo...
-->
</style>
```

- Vale, entendido.

- Si especificas una hoja de estilo CSS externa con `LINK` y otra incluida con `STYLE` se tienen en cuenta las reglas de las dos, por eso se llaman hojas *en cascada*. Cuando hay conflictos, para saber qué reglas de estilo predominan, hay unas reglas de cascada precisas definidas por la especificación. Hablaremos de eso en el tutorial siguiente.

- Ah, muy interesante.

- La tercera forma de especificar estilos es el atributo `style`, con el que puedes especificar reglas de estilo para un elemento en concreto. Puedes ver un ejemplo en la especificación, pero nosotros no lo vamos a utilizar.

- Me parece muy bien.

- Además, HTML ha heredado de sus versiones anteriores varios elementos para dar estilo: `TT`, `I`, `B`, `BIG`, `SMALL`, `FONT`, `BASEFONT`, `SUP`, `SUB` y `HR`. Algunos de ellos están desaprobados y otros no. Pero en cualquier caso nosotros tampoco los vamos a usar, porque lo apropiado es usar hojas de estilo. La única excepción es el elemento `HR`, que sirve para dibujar una línea separadora horizontal, siempre que lo consideremos como una separación estructural:

Elemento	`HR`
Modelo de contenido	VACIO

De todas formas podremos controlar la apariencia de la línea de división con hojas de estilo y en muchos casos querremos que no aparezca en los navegadores modernos. Bueno, ya está, vamos con el siguiente grupo de elementos.

- Genial, pues vamos.

Marcos

- El siguiente grupo de elementos son los relacionados con *marcos*. Los marcos en principio son una buena idea, porque permiten dividir la ventana en varias partes que son independientes entre sí, así puedes tener un menú fijo a la izquierda y a la derecha el contenido que cambia y se desplaza. Si estás viendo HTML con Clase con un navegador con soporte CSS2 como IE5 o NS6 verás un ejemplo de algo parecido.

- Pues sí, a mí me parece muy práctico.

- Sin embargo, tal y como están implementados, los marcos de HTML son horribles y no deberías usarlos.

- ¿Y por qué tú sí puedes?

- No, en HTML con Clase las subventanas están definidas con CSS, que es como debería ser, ya que, como sabes, la apariencia se especifica en las hojas de estilo, no en HTML.

- Sí, tienes razón...

- Léete este artículo sobre problemas de los marcos, y vamos a pasar al siguiente grupo.

- Estupendo.

Formularios

- El siguiente grupo está relacionado con los *formularios*. Los formularios dan a las páginas web un grado de interactividad que, aunque no es muy grande, es bastante práctico para muchas cosas.

- ¿Por ejemplo?

- Por ejemplo para hacer búsquedas por palabras en un motor de búsqueda, para comprar o hacer pedidos, para escribir mensajes al dueño de la web, para suscribirse a revistas electrónicas o listas de correo, etc.

- Entiendo.

- Si ves la sección de la especificación dedicada a los formularios, verás que son un pequeño mundo en sí mismos. Hay muchos tipos de controles de formulario y elementos auxiliares, y algunas cuestiones técnicas.

- Sí, la verdad es que es bastante.

- Por tanto, como las tablas y los objetos, los formularios se merecen un tutorial aparte. De momento te resumo que un formulario está formado por un control `FORM` dentro del cual se sitúan los controles del formulario: botones, controles de entrada de texto, casillas de verificación, menúes, etc.

Elemento	`FORM`
Modelo de contenido	uno o más elementos en bloque y/o elementos `SCRIPT`, pero no otros elementos `FORM`
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Normalmente los controles se insertan con un elemento `INPUT`, y según el valor del atributo `type` se obtiene un tipo de control u otro.

Elemento	`INPUT`
Modelo de contenido	VACIO

El elemento `INPUT` tiene muchos atributos. Según el tipo de control se usan unos u otros. ¿Vamos bien?

- Sí.

- Además existen otros controles como `BUTTON` (botón), `TEXTAREA` (un área grande rectangular para introducir texto), y `SELECT` (una lista

de opciones).

Elemento	BUTTON
Modelo de contenido	Cero o más elementos en bloque o en línea, excepto: elementos A, elementos FORM, elementos FIELDSET y controles de formulario
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Elemento	TEXTAREA
Modelo de contenido	Caracteres de texto y entidades de caracteres
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Elemento	SELECT
Modelo de contenido	Uno o más elementos OPTGROUP y/o OPTION
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Los elementos OPTION son cada una de las opciones seleccionables de una lista de opciones SELECT. El elemento OPTGROUP te permite agrupar opciones entre sí:

Elemento	OPTION
Modelo de contenido	Caracteres de texto y entidades de caracteres
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Elemento	OPTGROUP
Modelo de contenido	Uno o más elementos OPTION
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Tienes un ejemplo gráfico en la especificación.

- Muy bonito.

- Además puedes asociar un rótulo de texto con un control de formulario con el elemento LABEL:

Elemento	LABEL
Modelo de contenido	Cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

También puedes agrupar controles de formulario con el elemento FIELDSET y poner un rótulo al grupo de controles con el elemento LEGEND:

Elemento	FIELDSET

Modelo de contenido	Un elemento `LEGEND`, y a continuación cero o más elementos en bloque o en línea
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Elemento	`LEGEND`
Modelo de contenido	Cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Esos son todos los elementos de este grupo...

- ...que no son pocos...

- Todos los formularios tienen un botón de envío. Cuando el usuario lo pulsa, se envían los datos del formulario a un programa que se ejecuta en alguna computadora conectada a Internet. Esta aplicación está especificada por su URL con el atributo `action` del elemento `FORM`. Existen distintos métodos de enviar los datos, y en cada caso habrá que elegir el más apropiado. Pero esa y otras cuestiones las veremos como te dije en un futuro tutorial.

- Sí, lo vamos dejando todo para futuros tutoriales, pero bueno...

- Venga, vamos a por el siguiente grupo, que ya queda poco.

Scripts

- El siguiente grupo de elementos son los relacionados con los *scripts*, y lo siento, pero para esto no tengo traducción. Son como

programillas.

- Ah, programillas... ¿como por ejemplo?

- Los scripts que se usan en HTML son pequeños programas que se ejecutan en el ordenador del usuario. Tienen dos utilidades principales: la primera es validar formularios. Como te he dicho antes, cuanto el usuario pulsa el botón de enviar el formulario, los datos se envían al servidor para que los procese. Una vez procesados, el servidor envía la respuesta al usuario. Eso necesita tiempo, y si los datos enviados son incorrectos o insuficientes, es tiempo perdido. Por eso conviene comprobar la mayor cantidad posible de datos en el ordenador del usuario antes de enviar el formulario, y eso se puede hacer con scripts.

- Eso sí es una buena idea, ¿no?

- Sí, siempre y cuando los datos se validen en el ordenador del usuario, y además, en el servidor.

- ¿Y por qué hacer la comprobación dos veces?

- Porque no todos los navegadores tienen capacidad de ejecutar scripts, y algunos usuarios lo desactivan a propósito. Por tanto no puedes basarte en un script para que funcione una página.

- Entiendo. ¿Y por qué hay gente que desactiva los scritps?

- En primer lugar porque algunos navegadores tienen problemas de seguridad relacionados con JavaScript. No son problemas de JavaScript, sino de la implementación que hacen los navegadores de él. Eso le obliga a uno a actualizar su navegador periódicamente. Y otro motivo es porque con los scripts puedes abrir ventanas nuevas sin el permiso del usuario, lo cual fastidia bastante.

- Sí...

- La segunda utilidad de los scripts es crear HTML dinámico. Es decir, cambiar los estilos en tiempo real: visibilidad de elementos, cambios de posición, cambios de color, etc. Lo cual si se hace bien y

sin exagerar te puede ayudar a conseguir documentos más atractivos.

- ¿Y es fácil?

- Pues... sí y no. En primer lugar tienes que aprender un lenguaje de scripts. El más utilizado con diferencia es JavaScript. En segundo lugar para cada navegador tienes que programar rutinas diferentes para hacer las mismas cosas, lo cual te obliga a multiplicar tu tiempo de aprendizaje. A medida que se vayan implementando los estándares será mucho más sencillo. También tendremos un tutorial sobre HTML dinámico en el futuro.

- Ah, genial.

- En cualquier caso recuerda que no todo el mundo tiene JavaScript, así que la utilización de tu página no puede depender de que JavaScript esté activado. Tómate eso como otra regla básica de diseño web.

- De acuerdo, de acuerdo.

- Aquí tienes el elemento `SCRIPT`. Es parecido al `STYLE`: va en el elemento `HEAD` y en su atributo `type` se especifica el lenguaje de scripts, por ejemplo, `text/javascript`:

Elemento	`SCRIPT`
Modelo de contenido	sentencias de script (depende del lenguaje de scripts)
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Para tener en cuenta los navegadores sin soporte de scripts, existe el elemento `NOSCRIPT`. Si el navegador no ejecuta el script, sí debería mostrar el contenido del elemento `NOSCRIPT`, y viceversa:

Elemento	NOSCRIPT
Modelo de contenido	uno o más elementos en bloque
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

En teoría, lo ideal es que el elemento NOSCRIPT no te fuera nunca necesario. Por último, también se pueden incluir sentencias de script dentro de elementos particulares. Así puedes hacer que sucedan cosas cada vez que ocurren ciertos eventos, por ejemplo, cuando el documento termina de cargarse, cada vez que el ratón pasa por encima de un párrafo, etc. Pero aún tenemos cosas que aprender antes de meternos en estos berenjenales. ¿Acabamos ya?

- Sí, vamos a terminar.

Otros elementos

- Nos quedan cuatro o cinco elementos más para terminar el tutorial de hoy. Los dos primeros son los elementos INS y DEL, que se usan para marcar en el código HTML las partes que se han insertado y se han eliminado, junto con sus motivos. No es probable que lo utilices.

- Pues no, no se me habría ocurrido.

- Otro elemento especial es el elemento BDO, que está relacionado con la direccionalidad del texto. Te podría resultar interesante si mezclaras escritura occidental de izquierda a derecha con otros lenguajes donde se escribe de derecha a izquierda.

- Eeh... no, tampoco creo que lo utilice.

- Lo que suponía. Por último hay dos elementos muy útiles que vamos a usar bastante junto con las hojas de estilo: los elementos

`DIV` y `SPAN`. Nos permiten agrupar un conjunto de elementos y declarar un conjunto de reglas de estilo para el conjunto. Por ejemplo, el color de fondo de un conjunto de párrafos.

- ¿Y en qué se diferencian?

- `DIV` es un elemento en bloque, y `SPAN` es un elemento en línea:

Elemento	`DIV`
Modelo de contenido	cero o más elementos en bloque y/o en línea
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

Elemento	`SPAN`
Modelo de contenido	cero o más elementos en línea
Etiqueta inicial	obligatoria
Etiqueta final	obligatoria

En el siguiente tutorial veremos cómo podemos declarar reglas de estilo para elementos particulares. CSS tiene varios métodos para hacerlo. Hasta ahora lo que hemos hecho ha sido asignar estilos a cada tipo de elemento. Se puede restringir eso a los tipos de elementos que sean hijos de otros tipos de elemento. Y se puede restringir todavía más usando los atributos de HTML `class` e `id` que pueden tener todos los elementos de HTML. Pero todo eso lo veremos a fondo en el siguiente tutorial.

- Sí, mejor en el siguiente tutorial, porque yo ya no estoy para estos trotes.

- Lo comprendo perfectamente. Pero anímate, que ya hemos acabado. Hay algunos elementos más, pero sobre algunos ya hablaremos en el futuro y sobre otros no porque están desaprobados.

- Tienes que poner la lista de los elementos que son en bloque y en línea.

- Ah sí, aquí está:

Elementos en línea	Elementos en bloque
Datos de caracteres	P
Entidades de caracteres	H1, H2, H3, H4, H5, H6
TT, I, B, BIG, SMALL	UL, OL
EM, STRONG, DFN, CODE, SAMP	PRE
KBD, VAR, CITE, ABBR, ACRONYM	DL, DIV, NOSCRIPT, BLOCKQUOTE
A, IMG, OBJECT, BR, SCRIPT	FORM, HR, TABLE, FIELDSET, ADDRESS
MAP, Q, SUB, SUP, SPAN, BDO	
INPUT, SELECT, TEXTAREA, LABEL, BUTTON	

Ni se te ocurra aprendértelo de memoria. Es más fácil aplicar el sentido común.

- Ah sí, el sentido común, he oído hablar de él...

contacto

- Ahora lo que deberías hacer es investigar por tu cuenta. Ve navegando por ahí y ponte a mirar los códigos fuente de las páginas que visites. Te llevarás muchas sorpresas, así que si tienes alguna pregunta ya sabes dónde estoy. Y si quieres mandarme tus experimentos también.

- Muy bien.

- Pues venga, nos vemos en el próximo tutorial. Pásatelo bien.

- Vale, hasta otra.

Hojas de Estilo en Cascada, CSS1

La estructura de un documento HTML... otra vez

- Hola, hacía ya tiempo que no nos veíamos.

- Pues sí, pensaba que te habías olvidado de mí. Hoy hablamos de hojas de estilo ¿verdad?

- Sí, y en primer lugar vamos a hablar de la estructura de un documento HTML.

- ¿Otra vez?

- Sí, para manejar las hojas de estilo correctamente es indispensable comprender perfectamente cómo está estructurado un documento HTML.

- Muy bien.

- En un tutorial anterior te dije que a menudo se asemeja un documento HTML con un árbol.

- Si me acuerdo del dibujito.

- En realidad, a mí me gusta más imaginarlo como un montón de cajas que se contienen unas a otras. La caja mayor, equivaldría al elemento del documento, el elemento HTML, y a cada elemento le correspondería una caja más pequeña, así:


Un modelo de estructura de HTML: cada elemento es una caj que contiene a otras cajas, sucesivamente.

Aquí se ve claramente como unas cajas están contenidas en otras, es decir, cómo unos elementos están contenidos en otros. Si entiendes este modelo, habremos avanzado gran parte del camino.

- Yo creo que está bastante claro, sí.

- Bien. En jerga se habla de *elementos padres*, *elementos hijos* y *elementos hermanos*. En el modelo, un elemento padre corresponde a una caja que contiene otras cajas, que serían sus elementos hijos, y que serían hermanos entre sí. También se habla de *ascendientes* y *descendientes*. Los elementos ascendientes son los padres, los abuelos, etc., y los descendientes son los hijos, los nietos, etc.

- Vamos, que es como un árbol... de familia.

- Exactamente. Esto es importante porque una parte fundamental de las hojas de estilo es la *herencia*. Como sucede normalmente en la vida real, muchas de las propiedades de los padres son heredadas por sus hijos, pero no al revés.

- ¿Qué tipo de propiedades? ¿Las tierras, la casa?

- Las propiedades de estilo, hombre. Por ejemplo: el tipo de letra. Cuando especificas un tipo de letra para los contenidos de un elemento `P`, los elementos `EM` contenidos también tendrán en principio ese tipo de letra. Aunque que no todas las propiedades se heredan.

- Creo que más o menos lo entiendo.

- Bien, pues vamos a empezar ya.

Selectores

- Mira, esta es nuestra primera hoja de estilo, ¿te acuerdas?

```
body {
 color : rgb(151,0,0);
 background-color : rgb(255,201,0);
 font-family : Verdana, Arial, Helvetica, Helv, sans-serif;
}
```

- ¡Sí, qué tiempos aquellos!

- Esto es una de hoja estilo muy sencilla formada por un solo conjunto de reglas. La parte que va antes de las llaves, se llama *selector* (en este caso `body`), y sirve para seleccionar las partes del

documento a las que queremos que se apliquen las reglas de estilo que están entre llaves.

- Ya veo, entonces en este caso las reglas se aplican al elemento `BODY`, ¿verdad?

- Sí, eso. Pero CSS nos permite ser un poco más específicos con los selectores.

- ¿Qué quieres decir?

- Por ejemplo, con este conjunto de reglas:

```
EM { font-family : monospace }
```

todos los elementos `EM` del documento se representarían con una fuente no proporcional, es decir, con todos los caracteres de la misma anchura (siempre que una fuente así estuviera disponible, claro).

- Sí...

- Pero de esta otra forma podemos afinar un poco más:

```
OL EM { font-family : monospace }
```

Ahora la regla sólo se aplicará a los elementos `EM` que sean descendientes de algún elemento `OL`. Los elementos `EM` que no estén contenidos en un elemento `OL` no se verán afectados por esta regla.

- Anda, pues eso podría ser útil.

- Sí que lo es. A este tipo de selectores se les llama *selectores contextuales*. Otro ejemplo más:

```
TD OL EM { font-family : monospace }
```

se aplicará a los elementos `EM` que sean descendientes de algún elemento `OL`, que a su vez sean descendientes de algún elemento `TD`.

- Ya entiendo.

- Pero podemos ser aún mucho más específicos.

- ¿Cómo?

- Usando los atributos de HTML `id` y `class`. Con `id` y `class` podemos darle un poco más de significado a la estructura HTML de nuestro documento.

- No entiendo eso.

- Es sencillo, mira. En HTML, `P` es un párrafo, todos los `P` son párrafos. Pero para nosotros, desde el punto de vista estilístico, puede haber diferencias entre unos párrafos y otros. Por ejemplo, en un manual de instrucciones, la mayoría de los párrafos se dedicarán a explicar el funcionamiento de un aparato, pero en ocasiones pueden aparecer párrafos con advertencias, que queremos que aparezcan en cursiva. Así:

```
<h1>Instrucciones de seguridad</h1>
<p>Lea todas las instrucciones y conserve este manual de
instrucciones. Siga las advertencias e instrucciones existentes
sobre el monitor.</p>
<p class="advertencia">Atención: Nunca trate de abrir el
monitor usted mismo. En caso de avería diríjase siempre a
nuestro servicio técnico oficial.</p>
<p>Más instrucciones...</p>
<p class="advertencia">Atención: Otra advertencia...</p>
<p>Más instrucciones...</p>
```

- ¿Y cuál sería el selector para esos párrafos?

- El selector para todos los párrafos de la clase `"advertencia"` sería `p.advertencia`, es decir, separando con un punto el tipo de elemento, y la clase de elemento. Por ejempo:

```
p.advertencia { font-style : italic }
```

- ¿Es sencillo, no?

- Sí, pero es incluso algo más potente. Mira, imagina que en vez de un párrafo con una advertencia, ponemos una lista de advertencias. Pues sencillamente escribiríamos:

```
<ul class="advertencia">
<li>Antes de manipular el monitor desenchúfelo de la toma
de corriente.</li>
<li>Más advertencias...</li>
</ul>
```

Y ahora podríamos tener unas reglas para todos los elementos UL de clase advertencia:

```
ul.advertencia { font-style : italic }
```

O directamente, podríamos declarar unas reglas de estilo para cualquier elemento de clase advertencia:

```
.advertencia { font-style : italic }
```

- Ya veo, y eso se aplicaría cualquier tipo de elemento, siempre que tuviera `class="advertencia"`, ¿no?

- Sí. Así que el atributo `class` nos permite *clasificar* elementos.

- ¿Y el atributo `id`?

- El atributo `id` nos permite *identificar* elementos. Como has visto, un atributo `class` se puede aplicar a varios elementos. Pero el atributo `id` toma un valor único para cada elemento de un documento. Es decir, podemos dar a los elementos que queramos un identificador único que lo diferencie del resto, y eso es muy útil para hacer excepciones.

- ¿Por ejemplo?

- Por ejemplo:

```
<p class="advertencia" id="advertencia-final">Nota importante:
Si el usuario no sigue las instrucciones de seguridad
proporcionadas, se anulará automáticamente la garantía del
producto.</p>
```

Este párrafo se verá como el resto de las advertencias, pero además queremos aplicar unos estilos adicionales. Es tan sencillo como esto:

```
#advertencia-final { color : red }
```

- Ya veo, o sea que el carácter `#` se usa cuando el selector se refiere a un atributo `id`, ¿no?

- Exactamente. Otra cosa importante de los selectores es que se pueden agrupar, por ejemplo:

```
h1, .info h2, h3#notas { font-weight : bold }
```

indica que queremos que se escriban en negrita: los elementos `H1`, los elementos `H2` que tengan algún elemento de clase `"info"` entre sus ascendientes, y el elemento `H3` con identificador `"notas"`.

- ¡Increíble, creo que lo entiendo!

- Hay otro tipo de selectores, que se llaman *pseudo-clases* y *pseudo-elementos*, pero de esos hablaremos más tarde. Ahora

tenemos que hablar un poco de elementos en bloque y elementos en línea.

- Muy bien, vamos a ello.

Elementos en bloque y elementos en línea

- Ya hemos hablado en varias ocasiones de elementos en bloque y elementos en línea.

- Sí, siempre estás hablando de lo mismo...

- La diferencia entre unos y otros es fundamentalmente estilística o de presentación, porque los elementos en bloque normalmente se representan con un salto de línea antes y otro después, mientras que los elementos en línea forman parte de una línea.

- Sí, sí, lo sé.

- Lo que tienen en común entre sí es que son rectángulos, por ejemplo:

Un ejemplo de cuadros creados por elementos en bloque y e línea.

Cada rectángulo o *cuadro* tiene un área de *contenido* y opcionalmente un *borde*. Entre el borde y el contenido hay un área de *relleno* y entre el borde y los límites del cuadro hay un área de *margen*. El margen es lo que separa unos elementos en bloque de otros:

Un cuadro en bloque y sus dimensiones.

La *anchura del elemento* es la anchura del área del contenido, y la *anchura del cuadro* es la suma de las anchuras de elemento, relleno, borde y margen. Lo mismo con la altura: la *altura del cuadro* es la suma de las alturas de elemento, relleno, borde y margen. ¿Me sigues?

- Sí, creo que sí.

- La diferencia está en que, en general, para los elementos en bloque puedes especificar la anchura y la altura del elemento, mientras

que para los elementos en línea la anchura y la altura del elemento es en general la mínima para que quepa el elemento.

- Aaaah... mmh, no sé si te entiendo...

- Lo vamos a ver con una figura, así estará más claro:


Un cuadro en bloque con una altura arbitraria, y un cuadro e línea, cuya altura es la altura de una línea.

El modelo de cuadros es en realidad más complejo: hay *elementos flotantes*, *elementos objeto de lista*, *elementos reemplazados*... Te aconsejo que te leas el capítulo 4 de CSS1 (en español), y tendrás una visión completa del modelo de formato visual de CSS1.

- Bueno, intentaré sacar un poco de tiempo.

- Y el modelo de cuadros de CSS2 es aún más complejo, pero de eso hablaremos en su momento. Por ahora vamos a hablar un poco más de los elementos en bloque.

- Muy bien, lo que tú digas.

Modelo de cuadros CSS para elementos en bloque

- Dentro de los elementos en bloque se consideran, además de los elementos en bloque en sí, los elementos *objetos de lista* y los elementos *flotantes*.

- ¿Qué es un elemento flotante?

- El caso típico de un elemento flotante es una figura metida en un párrafo, a un lado. El texto rodea a la figura, ajustándose automáticamente a la anchura disponible, así:


Ejemplo de un objeto flotante en CSS. El texto se adapt automáticamente al ancho disponible.

- Ah, ya entiendo.

- En esta figura se representa un elemento en bloque (el elemento UL) y elementos objeto de lista (los elementos LI):

 Ejemplo de los cuadros creados por una lista no ordenada co dos objetos de lista.

Como hemos dicho antes, cada elemento en bloque define un cuadro, y los márgenes son los que fijan la distancia que queda entre los cuadros. Pues bien, los márgenes verticales de dos elementos en bloque adyacentes no se suman, sino que se reducen al mayor de los dos.

- No entiendo.

- Mira. El cuadro azul tiene un margen inferior de 4 centímetros. El cuadro rojo tiene un margen superior de 2 centímetros. Si colocamos el cuadro rojo a continuación del cuadro azul, la distancia que los separa es de 4 centímetros, que es el máximo de 4cm y 2cm, y no 6cm, que es lo que podría parecer lógico en un principio:

 Ejemplo de cómo se colapsan los márgenes verticales de dc cuadros adyacentes.

- Ajá...

- Se supone que así se obtienen resultados más acordes con lo que espera el diseñador.

- Ya.

- La especificación también habla de cómo calcular la anchura de los elementos en bloque, y la de sus márgenes, rellenos y bordes derechos e izquierdos, pero no dice nada de la altura. En principio la altura no es relevante, porque los elementos en bloque se van colocando unos a continuación de otros, y no hay mucho control sobre el formato vertical. Ten en cuenta que no todos los medios que acceden a la Web contemplan el concepto de "página". La especificación CSS2 es mucho más completa en ese sentido, y más compleja, porque hay muchas más posibilidades a la hora de controlar la posición exacta de cada cuadro, y de especificar reglas para medios concretos (pantallas, impresoras, sintetizadores de voz, etc.).

- Sí, pero de eso ya hablaremos en el futuro, ¿verdad?

- Verdad. De los elementos objeto de lista, debes saber que la única particularidad que tienen es que se representan con un marcador. Lo veremos con más detalles cuando hablemos de las propiedades de estilo. Y sobre los elementos flotantes, pues... la verdad es que es un pequeño lío...

- Uf, pues si para ti es un lío, imagina para mí...

- Lo mejor es saber que si un cuadro es flotante, el texto de los demás cuadros lo rodearán, respetando siempre los márgenes del elemento flotante, como se ve en esta figura:

 Ejemplo de cómo se conservan los márgenes de los cuadro flotantes.

Como ves, los márgenes del elemento flotante se suman a los márgenes del cuadro del párrafo, y no sucede como te he indicado antes para el margen superior.

- Mmm, ya veo...

- Hablemos un poco de los elementos en línea.

- Hablemos, hablemos...

Modelo de cuadros CSS para elementos en línea

- Los *elementos en línea* son elementos que ocupan una o varias líneas del texto de un elemento en bloque. Por ejemplo, un elemento `EM` será un elemento en línea. En una misma línea puede haber varios elementos en línea, cada uno de los cuales define su propio cuadro:

 Ejemplo de varios cuadros en línea en una misma línea.

Si un elemento no cabe en una línea, crea un cuadro diferente para cada línea en la que está presente:

 Ejemplo de cómo un cuadro en línea se divide en dos cuando elemento no cabe en una línea.

En general no podemos especificar la anchura y la altura de un elemento en línea, aunque sí sus márgenes, bordes y rellenos.

- ¿Por qué dices "en general"?

- Porque existe un tipo especial de elementos para los que sí se puede: los elementos *reemplazados*. Un elemento reemplazado puede ser un elemento en bloque o en línea. Lo que lo define es el hecho de que sus dimensiones en principio son desconocidas.

- Mmmm, ¿por ejemplo?

- Por ejemplo, el elemento `IMG`, si recuerdas del tutorial anterior, tiene esta forma:

```
<img src="imagen.png" alt="Texto alternativo para la imagen">
```

El navegador sustituirá el elemento por una imagen. La imagen tendrá una anchura y altura intrínsecas, pero en el código HTML esas dimensiones no vienen. Hasta que el elemento no es reemplazado por la imagen correspondiente, las dimensiones no se conocen.

- Aah, ya entiendo. Pero recuerdo que me dijiste que se puede poner la anchura y la altura con los atributos `width` y `height` del elemento `IMG`.

- Sí. Puedes especificar la anchura y la altura con esos parámetros, o puedes hacerlo en una hoja de estilo. Y puedes poner los valores reales del tamaño de la imagen, u otros valores más grandes o más pequeños. Y este es el caso en que para un elemento en línea (un `IMG` normalmente es un elemento en línea) se pueden especificar la anchura y la altura.

- Ajá. Sí, creo que te sigo.

- Lo que sí se puede especificar para los elementos en línea es la altura de las líneas. Existe un atributo que permite definir la altura de las líneas, o sea, la distancia que separa cada dos líneas de un párrafo. Lo veremos más a fondo después.

- Muy bien.

- Genial. Un detalle más antes de empezar con el repaso de las propiedades de estilo. El fondo sobre el que todo se representa, el *lienzo*, corresponde típicamente el elemento `BODY`. Esto quiere decir que para especificar por ejemplo el color de fondo, los márgenes, etc.

del documento, habrá que aplicar estas propiedades de estilo al elemento `BODY`.

- Entiendo.

- En CSS2 la noción del lienzo es conceptualmente más abstracta y suele dar lugar a quebraderos de cabeza. De momento, para nosotros, la anchura del lienzo es para que nos entendamos la anchura de la ventana del navegador, y la altura del lienzo es la mínima necesaria. ¿De acuerdo?

- Está bien...

- Supongo que ahora mismo tendrás un lío morrocotudo con todo lo de elementos en bloque y en línea...

- ¡Lo puedes jurar!

- Bueno, en cuanto tengas un poquito de experiencia te darás cuenta de que en cierto sentido la especificación CSS1 es muy simple. Unas párrafos van a continuación de los otros, y la altura de los párrafos casi siempre viene definida simplemente por sus contenidos y por la altura de sus líneas. Lo único que se escapa a este "flujo normal" son los elementos flotantes, que se pueden llevar a la izquierda o a la derecha, y a los que el resto del texto se adapta ocupando el ancho que dejan libre.

- Ya veo.

- Donde nos permite actuar CSS1 por tanto es: en los estilos de las letras (fuentes, negrita, cursiva, etc.), colores de texto y fondos, estilos de texto (separación entre palabras, alineación, altura de líneas...), propiedades de cuadros (márgenes, rellenos, bordes, tamaño), y estilos para las listas (tipo de marcador, tipo de lista, etc.). Si te parece vamos a ver las propiedades de CSS1 una a una.

- Ok, me parece bien.

Propiedades de fuente

- Empezamos con las propiedades de fuente. Se definen las siguientes propiedades:

- `font-family` : familia tipográfica (el nombre de la fuente)
- `font-style` : estilo de fuente (normal, itálica, inclinada)
- `font-variant` : variante (normal, versalitas)
- `font-weight` : peso (normal, negrita)
- `font-size` : tamaño de fuente
- `font` : una propiedad abreviada, que permite resumir todas las anteriores en una sola regla

- Ok.

- `font-family` toma como valores nombre de fuentes, en orden de preferencia. Si el nombre de la fuente contiene espacios, se entrecomilla. Al final de la lista conviene siempre poner un nombre de familia genérica que actúa como "red de seguridad", y que se refiere a un tipo general de fuente. Por ejemplo:

```
font-family : Georgia, "Times New Roman", Times, serif;
font-family : Verdana, Arial, Helvetica, sans-serif;
font-family : "Courier New", courier, monospace;
```

Las familias genéricas son: \'serif\' (p.ej. Times), \'sans-serif\' (p.ej. Arial), \'cursive\' (p.ej. Zapf-Chancery), \'fantasy\' (p.ej. Western) y \'monospace\' (p.ej. Courier).

- Bueno, vale...

- Seguimos. `font-style` permite especificar si queremos que la fuente esté en itálica. Puede tomar los valores `normal`, `italic` y `oblique`. Por ejemplo, ¿qué significa esto?

```
H1, H2, H3 { font-style: italic }
H1 EM { font-style: normal }
```

- Mmm, veamos, a ver si lo digo bien: que los contenidos de los elementos `H1`, `H2` y `H3` se representarán en itálica, excepto los contenidos de los elementos `EM` que sean descendientes de algún `H1`, que se verán normal.

- ¡Muy bien! Muy bien. `font-variant` puede tomar los valores `normal` y `small-caps` (versalitas).

- ¿Qué es eso?

- En una fuente con letras versalitas las letras minúsculas son similares a las mayúsculas, pero más pequeñas.

- Vale. Y con poner `font-variant : small-caps` vale, ¿no?

- Sí, si hay alguna fuente disponible en el sistema para ello sí. Si no, es posible que simplemente salgan todas las letras en mayúscula.

- Ok.

- `font-weight` puede tomar los valores `normal`, `bold` (negrita), `bolder` (más negrita que el padre), `lighter` (menos negrita que el padre), `100`, `200`, `300`, `400`, `500`, `600`, `700`, `800` ó `900`.

- Eso sí que es raro.

- Sí. El valor `400` equivale a peso `normal`, y el valor `bold` equivale a `700`. Lo habitual es poner simplemente `normal` o `bold`.

- Me parece bien...

- `font-size` permite especificar el tamaño de la fuente. Aquí hay bastantes opciones. Puede tomar los siguientes valores:

- un tamaño absoluto: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`;
- un tamaño relativo (al elemento padre): `larger` (más grande), `smaller` (más pequeño)
- una longitud, por ejemplo, `2cm` (2 centímetros), `3em` (3 veces el tamaño normal), etc.
- un porcentaje (del tamaño de fuente del elemento padre): por ejemplo, `150%` equivale a `1.5em`, o 1,5 veces el tamaño normal.

Cuando hagas una hoja de estilo para pantallas, conviene usar valores que le resulten cómodos al usuario, porque como sabrás leer en la pantalla cansa más que leer de un papel. En general, lo mejor es no especificar ningún tamaño de fuente, y si especificas algo, que sea una medida relativa al tamaño normal: un múltiplo de `em`, para que el usuario pueda escalar fácilmente los textos.

- ¿Qué es un `em`?

- Es un término tipográfico. Simplificando, es la altura de la fuente.

- Ah.

- En CSS, cuando hablamos de longitudes, hablamos de un número y su unidad. No puedes decir que algo mide 6. ¿6 qué? ¿6 centímetros, 6 metros?

- Hombre, claro.

- Las unidades que se pueden usar son las siguientes: `em` (la altura de la fuente), `ex` (la altura de la letra x), `px` (píxeles de pantalla), `in` (pulgadas, una pulgada son 2,54cm), `cm` (centímetros), `mm` (milímetros), `pt` (puntos, 72 puntos son una pulgada), y `pc` (picas, 1pc=12pt). Para impresoras y otros medios las unidades absolutas son apropiadas (`in`, `cm`, `mm`, `pt`, `pc`), pero para pantallas lo mejor es usar `em` o medidas porcentuales, ya que no se sabe a cuántos píxeles equivale por ejemplo un centímetro. La unidad `px` es un híbrido de medida absoluta y relativa, y de momento es aconsejable no utilizarla.

- Bueno, entonces no me quedan muchas opciones...

- La sintaxis es siempre el número y pegado a él la unidad. Se usa un punto decimal, no una coma decimal. Y cuando el número sea el cero, no hace falta unidad.

- Mmm, bastante lógico, sí.

- Bueno, pues seguimos. Finalmente, la propiedad `font` es un compendio de las anteriores. Aquí tienes un ejemplo completo:

```
font : italic small-caps bold 1.5em/3em Arial, sans-serif
```

Los tres primeros valores (`font-style`, `font-variant` y `font-weight`) son opcionales y pueden aparecer en cualquier orden. El cuarto (`font-size`) es obligatorio. El quinto (`line-height`, altura de línea) es opcional y si aparece debe estar precedido de una barra inclinada. Y al final se pone obligatoriamente la familia tipográfica (`font-family`). Otro ejemplo más simple:

```
font : 80% monospace
```

- Muy bien.

- Vamos al siguiente grupo de propiedades.

Propiedades de color y fondo

- El siguiente grupo son las propiedades de color y fondo, y son las siguientes:

- `color`: color del texto
- `background-color`: color del fondo
- `background-image`: imagen de fondo
- `background-repeat`: patrón de repetición de la imagen de fondo (si se repite, y cómo se repite)
- `background-attachment`: imagen fija o desplazable
- `background-position`: posición de la imagen de fondo
- `background`: propiedad abreviada para las propiedades de fondo.

La propiedad `color` es sencilla. Todas estas reglas significan lo mismo:

```
color : red;
color : rgb(255,0,0);
color : rgb(100%,0,0);
color : #ff0000;
color : #f00;
```

- ¿Todo eso significa lo mismo?

- Sí. Como ves hay cinco maneras de especificar un color en CSS:

1. Usando uno de los nombres de color predefinidos por CSS1: `aqua` (celeste), `black` (negro), `blue` (azul), `fuchsia` (fucsia), `gray` (gris), `green` (verde), `lime` (verde lima), `maroon` (marrón), `navy` (azul marino), `olive` (verde oliva), `purple` (púrpura), `red` (rojo), `silver` (gris claro), `teal` (verde azulado), `white` (blanco) y `yellow` (amarillo).
2. Especificando las componentes RGB (rojo, verde y azul) del color, de acuerdo a alguna de las cuatro notaciones siguientes:
   - `#rrggbb`, en hexadecimal, cada componente de `00` a `FF`. Por ejemplo `#4f6ac9 = RGB(4*16+15,6*16+10,12*16+9) = RGB(79,106,201)`
   - `#rgb` en hexadecimal. Por ejemplo `#369 = #336699 = RGB(51,102,153)`
   - `RGB(R,G,B)` en decimal, cada componente de 0 a 255. Por ejemplo, `RGB(255,255,255)`=blanco, `RGB(0,0,0)`=negro,

`RGB(0,0,255)`=azul

- `RGB(R%,G%,B%)` en porcentajes. Por ejemplo, `RGB(0,100%,0)`=verde

- ¡Uf, es un poco lío!

- Sí, ya le irás pillando el truco. Vamos con las propiedades de fondo. El color de fondo se especifica con `background-color` y es igual que antes:

```
background-color : valor de color
```

También puede tomar el valor:

```
background-color : transparent
```

o sea, ningún valor, que es el valor por defecto. Por ejemplo:

```
P { background-color : #ff0 }
```

- Veamos, eso sería `#ffff00`, máximo rojo, máximo verde, mínimo azul... mmm, no sé...

- Amarillo puro.

- Ah.

- Venga, esto deberías entenderlo, léete los artículos que te recomendé en el segundo tutorial.

- Bueeeno, vaaale.

- Seguimos. `background-image` permite especificar una imagen de fondo. Conviene especificar al mismo tiempo una imagen de fondo y un color de fondo, por si no se puede cargar la imagen:

```
body { background-image : url(ajedrez.gif) ; background-color :
white }
```

El valor por defecto es `none`, ningún fondo:

```
p { background-image : none }
```

- Muy bien.

- Con `background-repeat` decimos si la imagen de fondo se repite, y cómo lo hace. Los valores que puede tomar son:

- `repeat`: repetir horizontal y verticalmente
- `repeat-x`: repetir horizontalmente
- `repeat-y`: repetir verticalmente

- `no-repeat`: no repetir

  - Está claro...

  - Por ejemplo:

```
body {
 background-image : url(ajedrez.gif);
 background-color : white;
 background-repeat : repeat
}
```

  - Muy bonito.

  - `background-attachment` dice si la imagen está pegada al fondo, o si se mueve con el resto de los contenidos. En el primer caso el valor es `fixed`, en el segundo caso es `scroll`:

```
body {
 background-image : url(ajedrez.gif);
 background-color : white;
 background-repeat : repeat;
 background-attachment : fixed;
}
```

  - Ya me lo estoy imaginando...

  - Y `background-position` dice la posición inicial de la imagen de fondo. Son dos valores, y hay varias maneras de especificarlos, pero es un poco difícil de explicar. Lo mejor es que lo leas en la especificación. Si no lo entiendes no te preocupes, no es normal utilizar esto.

  - Muy bien, allí lo leeré.

  - Vale. Por último, `background` nos permite declarar todas las reglas de una tacada, así:

```
background : white url(ajedrez.gif) repeat fixed;
```

Todos los valores son opcionales y van en cualquier orden. Podrías poner:

```
background : red
```

  - Genial.

  - En realidad no es lo mismo `background-color : red` que `background : red`, pero no nos vamos a enrollar más.

- Vale, vamos al siguiente grupo.

Propiedades de texto

- El siguiente grupo son las propiedades de texto, y son las siguientes:

- `word-spacing`: espacio entre palabras
- `letter-spacing`: espacio entre letras
- `text-decoration`: decoración del texto
- `vertical-align`: alineación vertical
- `text-transform`: transformación del texto
- `text-align`: alineación del texto
- `text-indent`: sangría o indentación de la primera línea
- `line-height`: altura de línea

- Vaya, son unas cuantas...

- `word-spacing` puede tomar el valor `normal`, o una longitud que se añade al valor normal. Puede ser un valor negativo:

```
p { word-spacing : 1em }
```

- Muy bien.

- `letter-spacing` es lo mismo, puede tomar el valor `normal`, o una longitud que se añade al valor normal, y puede tomar un valor negativo:

```
h1 { letter-spacing : 1cm }
```

- Entiendo, muy fácil.

- `text-decoration` puede tomar el valor `none` (ninguna decoración), o una combinación de los siguientes, `underline` (subrayado), `overline` (una raya por encima), `line-through` (tachado) y `blink` (intermitente). Por ejemplo:

```
.antiguo { text-decoration : line-through }
```

- Vale, sigue.

- `vertical-align` dice cómo será la alineación vertical de un elemento en línea, y puede tomar cualquiera de estos valores (pero

sólo uno): `baseline`, `sub`, `super`, `top`, `text-top`, `middle`, `bottom`, `text-bottom`, o un porcentaje.

- ¡Hala, casi nada!

- Las descripciones las tienes en la especificación, y lo mejor es que lo leas allí.

- Muy bien.

- `text-transform` puede tomar uno de estos cuatro valores:

- `capitalize`: convierte en mayúscula el primer carácter de cada palabra
- `uppercase`: convierte en mayúsculas todas las letras del elemento
- `lowercase`: convierte en minúsculas todas las letras del elemento
- `none`: neutraliza el valor heredado

- Mmm, ok.

- `text-align` dice cómo alinear horizontalmente el texto contenido en un elemento en bloque, y puede vale `left` (justificar a la izquierda), `right` (justificar a la derecha), `center` (centrar) y `justify` (justificar a ambos márgenes). Por ejemplo:

```
blockquote.introduccion { text-align : center }
```

Eso haría que los contenidos de los elementos `BLOCKQUOTE` de clase `introduccion` aparecieran centrados en su cuadro.

- Ya entiendo.

- `text-indent` especifica el margen de la primera línea. Puede ser una longitud (por ejemplo, `text-indent : 2em`) o un porcentaje de la anchura del elemento padre.

- Mmm, creo que eso podría ser útil.

- Posiblemente. Y por último `line-height` establece la altura de línea, o sea, la distancia entre las líneas de base de dos líneas sucesivas. Puede tomar uno de estos cuatro valores: `normal`, un número, una longitud o un porcentaje del tamaño de la fuente del elemento.

- ¿Un número, sin unidad? ¿Qué indicaría un número?

- Es un multiplicador del tamaño de la fuente del elemento. Por ejemplo, las tres declaraciones siguientes son equivalentes:

```
div { line-height: 1.2; font-size: 1em }
div { line-height: 1.2em; font-size: 1em }
div { line-height: 120%; font-size: 1em }
```

- Sí, creo que lo entiendo.

- ¿Pasamos al siguiente grupo?

- Pasamos

Propiedades de cuadro

- El siguiente grupo son las propiedades de cuadro y son bastantes:

- para establecer los márgenes: `margin-top`, `margin-right`, `margin-bottom`, `margin-left`, `margin`
- para establecer el relleno: `padding-top`, `padding-right`, `padding-bottom`, `padding-left`, `padding`
- para establecer los bordes: `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`, `border-width`; `border-color`; `border-style`; `border-top`, `border-right`, `border-bottom`, `border-left`; `border`
- otras: `width`, `height`, `float`, `clear`

- ¡Uf, creo que aquí nos van a dar las uvas!

- Tranqui, a ver si lo resumimos bien. Las propiedades de margen (`margin-top`, `margin-right`, `margin-bottom`, `margin-left`) pueden tomar como valor una longitud, un porcentaje o el valor `auto` (automático). El porcentaje se refiere al margen del ascendiente en bloque más cercano. Se permiten valores negativos. Por ejemplo: `margin-left : 2em`

- Pues vale...

- En `margin` englobas las cuatro propiedades de margen, dando cuatro valores consecutivos, para el margen superior (*top*), derecho

(*right*), inferior (*bottom*) e izquierdo (*left*) respectivamente. Por ejemplo, `margin : 1em 2em 1em 3em`. Más información en la especificación.

- Ok, ok.

- Las propiedades de relleno (`padding-top`, `padding-right`, `padding-bottom`, `padding-left`) puede tomar como valor una longitud, o un porcentaje, que también se refiere al relleno del ascendiente en bloque más cercano. Aquí no se permiten valores negativos. En `padding` englobas los valores para los cuatro rellenos en el mismo orden que antes. ¿Ok?

- Ok.

- Por ejemplo:

```
p {
 background-color : yellow;
 padding : 1em;
 margin : 1em;
}
```

Aquí el párrafo y un área de `1em` a cada lado (el área de relleno) tienen fondo amarillo. El margen es de fondo transparente:

Ejemplo de cómo se colorea el color de fondo de un cuadro co relleno (el área de relleno se colorea, el margen transparenta fondo).

- Ajá, ya veo, o sea que el margen es siempre transparente, ¿no?

- Sí. Las propiedades de anchura de borde (`border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`) pueden tomar uno de los siguientes valores: `thin` (delgado), `medium` (mediano), `thick` (grueso) o una longitud. Con `border-width` especificas los cuatro a la vez, en el mismo orden que antes. Por ejemplo:

```
p {
 background-color : yellow;
 padding : 1em;
 margin : 1em;
 border-style : solid;
 border-width : thin thick thick thin
}
```

No te preocupes por lo de `border-style`, en seguida lo vemos.

Ejemplo del uso de la propiedad border-width.

- Muy bien, sigue.

- `border-color` es como `border-width`, pero en vez de anchuras, ponemos colores. El valor por defecto para el color de los bordes es el mismo que el de la propiedad `color` del elemento:

```
p {
 background-color : silver;
 padding : 1em;
 margin : 1em;
 border-width : medium thick thick medium;
 border-color : white gray gray white;
 border-style : solid;
}
```

Ejemplo del uso de la propiedad border-color.

- Qué bonito.

- `border-style` dice el estilo de los cuatro bordes, y puede tomar uno de estos valores: `none` (ninguno), `dotted` (punteado), `dashed` (a trazos), `solid` (continuo), `double` (doble), `groove` (canal), `ridge` (cresta), `inset` (bajorrelieve), o `outset` (altorrelieve).

Para los cuatro últimos tienes que dar un valor a la propiedad `color` si quieres ver el efecto. Por ejemplo:

```
p {
 background-color : silver;
 color : maroon;
 padding : 1em;
 margin : 1em;
 border-style : inset
}
```

Ejemplo del uso de la propiedad border-style.

Las propiedades `border-top`, `border-right`, `border-bottom` y `border-left` permiten especificar, para cada borde, su anchura, su color y su estilo:

```
p {
 border-top : 3px #000 dashed;
 border-bottom : 3px #000 dashed;
 border-right : 3px #000 double;
```

```
 border-left : 3px #000 double;
}
```

 Ejemplo del uso de las propiedades border-top, border-botton border-right y border-left.

- Muy bien.

- Y la propiedad `border` es un resumen de las cuatro anteriores, que permite especificar, para los cuatro bordes, la anchura, el color y el estilo:

```
p { border : 3px black double }
```

- Je je, mucho más sencillo.

- Finalmente, `width` y `height` especifican la altura de un elemento en bloque o un elemento reemplazado, como una imagen. Pueden tomar el valor `auto`, una longitud, y `width` también puede ser un porcentaje de la anchura del elemento padre. Por ejemplo:

```
IMG.icono {
 width : 100px;
 height : 100px;
}
div.nota { width : 50% }
```

Un par de comentarios respecto a estas propiedades.

- A ver...

- Si recuerdas, te dije que la anchura del cuadro de un elemento en bloque es: la anchura del contenido, más la anchura de relleno, borde y márgenes.

- Sí.

- Por otra parte, la anchura del cuadro de un elemento es igual a la anchura del contenido de su elemento padre, por ejemplo:

 Ejemplo en que se ve cómo la anchura del cuadro de u elemento en bloque es igual a la anchura del contenido de element padre.

- Mmmm, sí.

- Esto quiere decir que los valores que demos a los márgenes, bordes y rellenos derecho e izquierdo, más la anchura del elemento,

deben ser igual a la anchura del cuadro del elemento, no pueden ser cualesquiera.

- Claro...

- Por eso CSS1 define unas reglas para que esto siempre se cumpla. Las puedes encontrar en la especificación. Al principio parecen un lío pero en realidad son bastante lógicas.

- Bueno, vale.

- Y con respecto a la altura, en CSS1 es más útil para elementos reemplazados que para elementos en bloque, porque no es necesario que el navegador tenga en cuenta la altura para elementos en bloque.

- Aaaah...

- Lo propiedad `float` indica que un elemento flota a la izquierda (`float:left`), a la derecha (`float:right`) o que no es un objeto flotante (`float:none`), por ejemplo:

```
img.figura {
 float : right;
 margin-left : 1em;
 margin-right : 0;
}
```

- Entiendo.

- Y por último la propiedad `clear` indica si un elemento puede tener objetos flotantes a sus lados. Con `clear:left`, el elemento se mueve hacia abajo hasta que quede por debajo de un hipotético objeto que flote a la izquierda. Con `clear:right` lo mismo pero a la derecha. Con `clear:both` no se permite que haya un objeto flotante a ningún lado, y con `clear:none` se permiten objetos flotantes a ambos lados:

```
h1 { clear : both }
```

- Ya entiendo.

- Estupendo, pues pasamos al último grupo.

- Genial.

Propiedades de clasificación

- El último grupo son las propiedades de clasificación, que nos permiten especificar cómo se comporta el elemento. Son las siguientes: `display`, `white-space`, `list-style-type`, `list-style-image`, `list-style-position` y `list-style`.

- Muy bien, empieza.

- `display` dice si un elemento es en bloque (`display:block`), en línea (`display:inline`), si es un objeto de lista (`display:list-item`) o si no debe ser representado (`display:none`).

- No entiendo, ¿esto quiere decir que puedo decir por ejemplo `em { display : block }`?

- Sí, lo que eso querría decir es que los elementos `EM`, en vez de abrir un cuadro en la misma línea, abrirían un nuevo cuadro en bloque. Pero lo mejor es no hacer estas cosas a menos que sea absolutamente necesario.

- Ok, muy bien.

- `white-space` es una propiedad que dice cómo tratar a los espacios contenidos en un elemento en bloque. Con el valor `normal`, varios espacios seguidos se convierten en uno solo, que es a lo que estamos acostumbrados. Con el valor `pre` obtienes el comportamiento del elemento de HTML `PRE`, en el que todos los espacios se conservan. Y con el valor `nowrap` sólo se producen saltos de línea donde haya elementos `BR`.

- Mmmm, muy bien.

- Bien, y para finalizar, una serie de propiedades que se aplican a elementos con `display:list-item`. La propiedad `list-style-type` dice el tipo de estilo de numeración de la lista. Puede ser: `disc` (un círculo relleno), `circle` (un círculo hueco), `square` (un cuadrado), `decimal` (números), `lower-roman` (números romanos en minúscula), `upper-roman` (números romanos en mayúscula), `lower-alpha` (letras minúsculas), `upper-alpha` (letras mayúsculas) o `none` (ningún marcador).

- Muy bien.

- A no ser que pongamos una imagen con `list-style-image`:

```
ul { list-style-image: url(imagenes/flecha.gif) }
```

- Anda, mira qué bien.

- La propiedad `list-style-position` indica el tipo de presentación de la lista, según dónde se coloque el marcador, y puede tomar los valores `inside` (interior) y `outside` (exterior):

Ejemplo del uso de la propiedad list-style-position.

Y por último la propiedad `list-style` es un resumen de las anteriores, y puedes indicar al mismo tiempo: el tipo de marcador, el estilo de posición y la imagen del marcador. Si la imagen no está disponible, se colocará un marcador del tipo especificado:

```
ul { list-style : url(imagenes/flecha.gif) circle }
```

- Estupendo, bien pensado.

- Muy bien, estamos a punto de terminar, pero tenemos que hablar aún de dos cosas importantes: las *pseudo-clases* y los *pseudo-elementos*, y en especial del *mecanismo de cascada*.

- Venga, vamos.

Pseudo-clases y pseudo-elementos

- Como hemos visto, los selectores se refieren a partes estructurales del documento: un tipo de elementos, los elementos que son descendientes de otros, los elementos de una clase, etc. Pero a veces es útil seleccionar elementos no sólo por su papel estructural, sino también por su presentación.

- A ver, explícate un poco.

- Por ejemplo, para seleccionar un elemento `A`, nos bastaría con el selector `A`, pero ¿qué pasa si quiero dar una apariencia diferente a los vínculos visitados y a los que aún no han sido visitados? Otro ejemplo, ¿cómo selecciono la primera línea de un párrafo, si hasta que no está representado no sé cuál es la primera línea?

- Sí, ya veo a qué te refieres.

- En CSS1 existen tres pseudo-clases para los elementos `A` que tengan un atributo `href`, que son `link` (no visitados), `visited` (visitados) y `active` (activos, por ejemplo, mientras el usuario está haciendo clic con el ratón sobre el vínculo). La manera de representar estas pseudo-clases es la siguiente:

```
A:link { color: blue }
A:visited { color: red }
A:active { color: lime }
```

Es decir, con dos puntos en lugar de uno solo.

- Ah, qué cosas.

- Es importante que `A:active` sea la última que declares por cuestiones de herencia, pero de eso hablaremos después.

- Vale.

- Hay otra pseudo-clase definida en CSS2 que a todo el mundo le gusta mucho, la pseudo-clase `hover`, que es para cuando el ratón pasa por encima del vínculo, y se debe poner justo antes de `A:active`:

```
A:link { color: blue }
A:visited { color: red }
A:hover { color : yellow }
A:active { color: lime }
```

- Muy bien.

- Los pseudo-elementos definidos en CSS1 son el correspondiente a la primera línea de un párrafo, y el correspondiente a la primera letra de un párrafo. Para hacer referencia a la primera línea de un párrafo, se hace lo siguiente:

```
P:first-line { font-weight : bold }
```

- Ya veo, en este caso la primera línea saldría en negrita, ¿no?

- Sí. Y para acceder a las propiedades de la primera letra, `P:first-letter`, por ejemplo:

```
p              { font-size: 1em; line-height: 1.2em; text-align
: justify }
p:first-letter { font-size: 200%; float: left; font-weight:
```

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

 Ejemplo del uso los pseudo-elementos tipográficos de CSS1.

- ¡Eh, eso está genial!

- Bien, pues vamos a hablar del mecanismo de cascada y acabar el tutorial.

- Muy bien.

La cascada

- Una de las características fundamentales de las hojas de estilo, es que se pueden combinar.

- ¿Qué quiere decir eso?

- Pues que puedes aplicar a un mismo documento HTML más de una hoja de estilo. Por ejemplo, una empresa puede tener una hoja de estilo común, y cada departamento de la empresa su propia hoja de estilo.

- Sí, podría ser útil.

- A eso hay que añadirle que cada usuario puede definir su propia hoja de estilo y aplicarla también el mismo documento.

- Pues qué lío ¿no? ¿Qué pasa si aplicamos al mismo elemento propiedades diferentes, o incluso opuestas?

- Sí, y no sólo eso, porque como te dije al principio, algunas propiedades se heredan de padres a hijos, lo cual complica aún más las cosas. Por eso CSS1 define unas reglas precisas para determinar cuáles son los valores que se aplican finalmente a las propiedades de estilo de cada elemento. A estas reglas se les llama el *mecanismo de cascada*.

- Pues a ver, enséñame ese mecanismo.

- Es un poco complicado, y voy a intentar explicártelo a grandes rasgos. Como siempre, la especificación contiene la descripción completa.

- Sí, venga, no seas tan vago y desembucha.

- Las hojas de estilo se dividen en dos grupos: *hojas del autor*, y *hoja del lector*. En CSS1, las reglas del autor tienen más peso que las del lector, a menos que el lector declare sus reglas como "importantes", así, por ejemplo:

```
body { font-size : 2em !important }
```

- ¿Y el autor no puede declarar sus reglas importantes también?

- Sí, y en ese caso sus reglas prevalecen sobre las del lector. Pero esto es así en CSS1, en CSS2 ha cambiado, y una regla importante del lector siempre tiene preferencia.

- Bueno, a mí me parece más lógico.

- Las hojas del autor se pueden importar básicamente de dos maneras: con varios elementos `LINK` desde el documento HTML, y con reglas `@import` en la hoja de estilo. Por ejemplo, una hoja de estilo podría ser así:

```
@import url(empresa.css);
@import url(dep-comercial.css);
h1 { font-style : italic }
```

Esto quiere decir que primero se cargan las reglas de `empresa.css`, a continuación las de `dep-comercial.css` y finalmente se añade la regla de esta hoja. ¿Lo ves?

- Sí.

- Pues bien, para cada propiedad de cada elemento, el valor especificado por la última regla es el que tiene preferencia, es como si anulara a los anteriores. Si por ejemplo en `dep-comercial.css` se hubiera especificado:

```
h1 { font-family : monospace; font-color : blue; font-style :
normal }
```

los valores de las propiedades de estilo resultantes para los `H1` serían:

```
h1 { font-family : monospace; font-color : blue; font-style :
italic }
```

- Ya entiendo.

- Estupendo. La regla `@import` debe ir obligatoriamente al principio de una hoja de estilo. Las reglas que empiezan con el símbolo `@` se llaman *reglas tipo arroba* (*at-rules* en inglés), y en CSS2 además de `@import` existen algunas más.

- Muy bien.

- Visto eso, veamos cómo se resuelven los conflictos. Para ello, CSS se basa en dos conceptos: el *orden en la cascada* (como acabamos de ver), y la *especificidad*. Por ejemplo, imagina que, en el caso anterior añadimos una nueva declaración:

```
@import url(empresa.css);
@import url(dep-comercial.css);
div.resumen h1 { font-style : normal }
h1 { font-style : italic }
```

¿Qué valor de `font-style` le darías a los `H1` que aparecen dentro de un `DIV` de clase `resumen`?

- Pues, no sé, en principio `normal`, pero después aparece la regla `font-style:italic` para todos los `H1`... no sé.

- Aplicaríamos `font-style:normal`, porque los selectores más específicos prevalecen sobre los más generales. Solamente en el caso en que dos selectores sean igual de específicos, se imponen las reglas del último declarado.

- Ya, ¿y cómo se mide la especificidad?, porque eso parece un concepto un poco abstracto...

- El concepto de especificidad está definido en la especificación. Cuentas el número de atributos `id` en el selector, por ejemplo "*a*"; cuentas el número de atributos `class` (y pseudo-clases), por ejemplo "*b*"; y el número de nombres de elementos (y pseudo-elementos), por ejemplo "*c*". Concatenando los tres números obtienes la especificidad "*abc*". Estos son algunos ejemplos sacados de la especificación CSS1:

```
LI            {...}  /* a=0 b=0 c=1 -> especificidad =   1 */
UL LI         {...}  /* a=0 b=0 c=2 -> especificidad =   2 */
UL OL LI      {...}  /* a=0 b=0 c=3 -> especificidad =   3 */
LI.red        {...}  /* a=0 b=1 c=1 -> especificidad =  11 */
```

```
UL OL LI.red  {...}  /* a=0 b=1 c=3 -> especificidad =  13 */
#x34y         {...}  /* a=1 b=0 c=0 -> especificidad = 100 */
```

- Vaya, ya veo que han atado todos los cabos...

- Por cierto, esos asteriscos, antes de que se me olvide, son la forma de poner comentarios en las hojas de estilo. No habíamos hablado de esto y es importante.

- Tsk, tsk...

- Se pueden poner comentarios en un documento HTML de la siguiente manera:

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

El comentario no aparecerá representado, pero te puede ser útil cuando repases tus documentos días o meses después de haberlos creado. En CSS los comentarios son diferentes:

```
/* Esto es un comentario CSS */
```

Sólo es eso, es muy sencillo y son útiles para no volverte más loco de lo necesario.

- Demasiado tarde para eso...

- Bueno, nunca es tarde para volverse más loco. En realidad todo esto de la cascada es más complicado de lo que te he explicado, y además en CSS2 las cosas varían ligeramente. Por no decir que la implementación que hacen los navegadores actuales de CSS no es precisamente perfecta...

- Calla, calla, no me asustes más.

- Bueno, creo que es el momento de ver unos cuantos ejemplos de HTML 4 y CSS1 auténtico y de verdad, ¿te parece?

- ¡Por supuesto!

Ejemplos

Muy bien, veamos algunos ejemplos. Recuerda que CSS1 no nos da muchas opciones, simplemente nos ayuda a adornar el documento. Lo que vamos a ver aquí sería un sueño para los primeros desarrolladores de HTML, pero con el posicionamiento CSS2 del que

hablaremos en el futuro se pueden conseguir cosas mucho más avanzadas.

- Hala, hala, no te enrolles tanto y enséñame esos ejemplos.

- Bueno, bueno, pero espera. Es aconsejable que veas estos ejemplos con Netscape 6.1 o superior, Opera 5 o superior o Mozilla, que son los navegadores que mejor implementan CSS1. Explorer 5.0 o superior dan resultados aceptables, aunque tienen muchos errores. Otros navegadores, como Navigator 4 o inferiores o Explorer 4 o inferiores dan resultados bastante malos. Algunos de los ejemplos utilizan algunos trucos para que las páginas sean compatibles con ellos, pero...

- Sí, sí, lo sé, ya hablaremos de esto en el futuro.

- Je, je, sí. Bueno, aquí tienes una lista de ejemplos:

- Esta es una de las primeras páginas que hizo uso real de CSS1: http://www.danielgreene.com/style.html, realizada por Daniel Greene. Actualmente no está en HTML 4, sino XHTML1.0, pero básicamente son lo mismo.
- Aquí tienes una página muy sencilla, pero muy elegante, de la Oxford Computer Gaming Society: http://users.ox.ac.uk/~gaming/.
- Otra página del mismo estilo, también en HTML 4.01 estricto y CSS1, es esta: http://www.geocities.com/marcoschmidt.geo/java.html
- Esto son unas pruebas mías: ejemplo6.html
- Esta página es bastante bonita en mi opinión: http://www.henkhaverhoek.nl/
- Una de las páginas "tradicionales", y que contiene un montón de información relacionada con hojas de estilo (en inglés, sorry), es esta del CSS Pointers Group: http://css.nu/pointers/
- Esto es de Jan Roland Eriksson, que es una de las personas que más sabe de esto: http://member.newsguy.com/~jrexon/
- Otro ejemplo, y una página muy interesante es ésta de Todd Fahrner, otro gran especialista: http://style.cleverchimp.com/.
- Esta página es de Rijk van Geijtenbeek (ni que decir tiene, otro que sabe mucho): http://rijk.op.het.net/info/niwo/
- En http://www.w3.org/StyleSheets/Core/ puedes ver una colección de hojas de estilo básicas del W3C, que se pueden usar libremente y sin necesidad de conocer CSS.

Bueno, pues yo creo que es suficiente. CSS es a la vez sencillo y complejo. Debes recordar siempre que en la Web no controlas, sólo sugieres, y que CSS no sirve para controlar, sólo para sugerir. Crea siempre en primer lugar tu página HTML, con su estructura lógica correcta, y a continuación, encima de eso, crea una hoja de estilo que le dé la apariencia que tú quieras, con las limitaciones, pero también con la flexibilidad, del medio.

- Bueno, pero si quieres que te sea sincero, creo que tengo el síndrome del escritor. Hemos visto todo el HTML y todo el CSS1, pero ahora no es tan fácil sentarse frente a una pantalla en blanco y empezar a escribir...

- Ya lo sé, pero tú empieza por lo fácil y ve experimentando, en un par de meses o tres verás como dominas bastante todo esto. Y cuando quieras preguntar, pregunta. La lista de correo está para eso.

- Vale, vale, ojalá tengas razón.

- Bueno, el próximo tutorial va a tratar sobre cómo subir un sitio a la Web con FTP, hablaremos de caracteres acentuados, y veremos algunos truquitos para ocultar las hojas de estilo a los navegadores antiguos que no las entienden bien.

- Estupendo, nos vemos entonces, ¡y no tardes tanto como esta vez!

- ¡Hago lo que puedo! Hasta pronto.

- Adiós, adiós.

Consejos antes de publicar la página

Cómo subir las páginas a la Web

- ¡Hola!

- Hola, ¿te gusta mi barba?

- ¿Cómo dices?

- Me creció mientras te esperaba...

- Bueno, bueno, perdona. Ya te dije que hago lo que puedo... ¿Y tú has hecho algo? No me has enviado ningún ejemplo.

- Estooo, será mejor que empecemos.

- Sí, claro. Vamos a empezar hablando sobre cómo subir tu sitio a Internet. Después hablaremos de un par de problemillas que suelen aparecer al publicar las páginas web y cómo resolverlos. Esta vez será un tutorial cortito.

- Ok, me parece muy bien.

- Naturalmente, lo primero que necesitas es un espacio para subir tus páginas.

- Naturalmente. ¿Y dónde lo encuentro?

- Hay muchos servicios de alojamiento gratuito. Normalmente a cambio ponen publicidad en tu página, para poder costear el servicio. Estos anuncios no tienen por qué coincidir con la temática o los intereses de tu web, y no tienes control sobre ellos. Al final resulta que todos tus esfuerzos por conseguir visitas (¡que no son pocos!) los aprovecha mucho más la compañía que te da el hosting que tú, y con el precio que tiene el alojamiento web en realidad, realmente no vale la pena.

- Muy bien.

- La manera más fácil y más cómoda de subir tus ficheros a tu espacio web es con FTP. ¿Te acuerdas del FTP, no?

- Sí, protocolo de transferencia de ficheros. Para transferir ficheros, ¿no?

- Sí, muy bien. Te aconsejo que elijas un servicio que te permita subir tus archivos por FTP. Entonces, asumiremos que tienes acceso por FTP.

- Vale, ¿y eso qué significa?

- A ver cómo te lo explico... ¿Tú sabes qué es un servidor?

- ¿Un servidor de Internet? Es una computadora conectada a Internet, ¿no?

- Sí, una máquina especial conectada permanente a Internet. En cierto modo, se llama servidor porque está siempre a tu servicio. Tú le pides una página web y él te la da.

- Quién lo habría dicho...

- Sí. Entonces un servidor FTP es una máquina que te permite subir tus ficheros a Internet. Todo lo que necesitas es un nombre de usuario y una contraseña para poder conectarte. Y eso te lo da tu proveedor de hosting.

- Ah, vale.

- Ahora necesitas un cliente de FTP, es decir, un programa que te permita conectarte con el servidor y comenzar a transferir ficheros.

- Lógico. ¿De dónde lo saco?

- Lo más probable es que tu sistema operativo ya venga con uno (aunque casi nadie lo sabe). Pero será mejor uno que sea más sencillo de utilizar. Para Windows te recomiendo FileZilla. Es gratis y está muy bien. También es probable que tu editor de páginas web te permita realizar transferencias por FTP.

- Bueno, me bajaré el FileZilla ese.

- Buena idea. Ahora es muy sencillo. Configuras en el programa tu cuenta, te conectas y transfieres tus archivos desde tu computadora al servidor (o al revés), pulsando en las flechitas. Hay varios tutoriales sobre el uso de FileZilla en la red. Léete alguno, ¿lo harás?

- Sí, sí, lo prometo.

- Bien, un par de consejos sobre el FTP:

- Es **muy** recomendable que los nombres de tus archivos sólo tengan letras minúsculas, o números, nada de espacios ni símbolos raros ni eñes ni letras con acentos. Eso te hará la vida muuucho más fácil, créeme.
- Existen dos maneras de transferir ficheros por FTP: en modo ASCII o en modo binario. El modo ASCII es para ficheros de texto, por ejemplo, documentos HTML, hojas de estilo CSS, scripts de Perl, etc. En general, todo lo que haya sido creado con un editor de texto. El modo binario es para todos los demás ficheros, por ejemplo, imágenes, ejecutables, animaciones, archivos comprimidos, etc. Normalmente los clientes de FTP suponen cuál es el modo en que tienen que trasferir el fichero según su extensión, y lo hacen automáticamente, pero en todo caso si te encuentras con problemas podrían estar causados por esto. (Por ejemplo, un error muy típico es subir un script CGI Perl en modo binario desde Windows.)

- De acuerdo, no sé de qué estás hablando pero lo tendré en cuenta.

- ¡Ay, paciencia! ¿Qué no entiendes?

- Eso de "subir un script CGI Perl en modo binario desde Windows" te ha quedado muy esotérico...

- Bah, no te preocupes, ya te lo explicaré cuando hablemos de formularios.

- Ok, ok...

- Muy bien, pues eso es todo. Realmente no tiene mucho secreto.

- Bueno, eso espero. Supongo que será cuestión de acostumbrarse, como todo.

- Sí, claro. Bueno, ¿seguimos?

- Seguimos.

Acentos y símbolos especiales

- Bien. Al subir tus páginas a la Web es posible que te encuentres con un pequeño problema.

- ¡No me digas! ¿Cuál?

- Que desaparezcan todos los acentos. A veces pasa. La ñ se convierte en q, la í se convierte en m, el signo ¿ se convierte en ?, etc. Sería un poco largo de explicar ahora, tiene algo que ver con las codificaciones de caracteres, y todo eso. De todas formas, lo importante es que tiene fácil arreglo.

- Ah, bueno, entonces...

- ¿Te acuerdas de cuando hablamos sobre referencias a entidades de caracteres?

- Mmmm, la verdad es que no...

- Sí hombre, lo vimos en el tutorial cuarto. Son códigos especiales que empiezan con un & y terminan con un punto y coma, y te puse como ejemplo el código para el símbolo del copyright, ©

- Ah, sí, es verdad.

- Bien, pues con estos códigos podemos poner caracteres acentuados y otros símbolos especiales sin miedo a que desaparezcan al subir la página a la Web.

- ¿Quieres decir que hay un código para cada carácter acentuado?

- Sí, hay muchos códigos. Puedes ver la lista completa en la especificación de HTML. Pero si escribes en castellano con estos te valdrán:

á	á	**Á**	Á
é	é	**É**	É
í	í	**Í**	Í
ó	ó	**Ó**	Ó
ú	ú	**Ú**	Ú
ñ	ñ	**Ñ**	Ñ
¿	¿	**¡**	¡
ü	ü	**Ü**	Ü
<	<	**>**	>
&	&		

- O sea que cada vez que quiera escribir á tengo que poner á ¿no?

- Bueno, en teoría no debería ser necesario, aunque como te he dicho es la mejor manera de asegurarte de que tu documento se verá lo mejor posible. De todos modos lo recomendable, si ves que tu servidor se come los acentos, es convertir los caracteres a sus referencias antes de subir el documento a la Web. O sea, escribes normalmente, conviertes los caracteres con el editor (con la función de reemplazar) y entonces lo subes.

- Ajá, entiendo.

- Además habría que decirle al navegador qué codificación de caracteres estamos usando, que es nuestro casos erá la ISO-8859-1...

- ¡Alto ahí! Tradúceme eso al castellano.

- Es algo un poco difícil de explicar, y es un poco lioso. ¿De verdad quieres que te lo explique?

- Sí, por favor.

- Muy bien. En primer lugar, existe una cosa llamada *Conjunto Universal de Caracteres*, o en inglés *Universal Character Set* (UCS). En teoría, contiene todos los posibles caracteres que se pueden escribir en todos los idiomas.

- Impresionante.

- ¿Verdad que sí? Bueno, en segundo lugar, existe otra cosa que se llama *Conjunto de Caracteres del HTML*, que son todos los caracteres que entiende un intérprete de HTML. Pues bien, resulta que el conjunto de caracteres del HTML es igual que el conjunto de universal caracteres.

- Entonces, ¿podría escribir mi documento HTML en hebreo, o en chino?

- Sí, perfectamente, pero antes tienes que avisar al navegador de que vas a utilizar cosas raras. Tienes que elegir una codificación de caracteres.

- Pero ¿por qué? Si me acabas de decir que el HTML lo entiende todo.

- Como sabes, la memoria de los ordenadores sólo acepta números. De modo que para almacenar la letra á, por ejemplo, lo que guarda él es el número 0225, que es el código de la letra á.

- Ya me estoy perdiendo, pero bueno. O sea, que cada letra corresponde a un número, ¿no?.

- Sí. Pero resulta que en el alfabeto griego, el número 0225 corresponde a la letra alpha. Entonces, si tú escribes tu documento en griego, y quieres que salgan letras griegas, tienes que decir que has utilizado la codificación griega de caracteres. Así, cuando al agente de usuario le llega el código 0225, no lo transforma en la letra á, sino en la letra griega alpha.

- O sea, que la codificación de caracteres le dice al agente de usuario a qué letra del conjunto universal de caracteres corresponde

cada código numérico.

- Exactamente. En nuestro caso, como nosotros escribimos en castellano, y tenemos acentos, eñes y toda la pesca, tenemos que utilizar la codificación ISO-8859-1.

- ¿Y cuál es la codificación griega?

- La ISO-8859-7. ¿Y tú para qué quieres la codificación griega?

- No, nada, pura curiosidad.

- Bueno, el caso es que existen varias maneras de declarar la codificación, algunas mejores que otras. Lo mejor es configurar al servidor para que sea él el que anuncie la codificación usada. De hecho es muy probable que ya esté configurado para ello. Pero para estar completamente seguros, y como eso probablemente no podrás hacerlo tú ahora, lo puedes hacer en HTML sin mucho peligro. Simplemente has de poner este elemento `META` en el `HEAD` del documento, justo después de la etiqueta `<head>`:

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

- ¡Qué cosas más raras me haces hacer!

- No es culpa mía. Bueno, como ves la cuestión de los caracteres especiales en la Web es bastante compleja. De momento nosotros nos vamos a parar aquí y no vamos a complicar las cosas más. ¿Seguimos?

- Sigamos, sigamos.

Ocultar las hojas de estilo a los navegadores antiguos

- Muy bien, para terminar, vamos a hablar un poco de otro problema con el que se pueden encontrar los que visiten nuestras páginas.

- ¿Y cuál es ese otro problema?

- Las hojas de estilo. En realidad, el problema son los navegadores que no implementan bien las hojas de estilo. Te dije en el tutorial anterior que algunos navegadores dan resultados solamente aceptables o muy malos, y que existían algunos trucos para ocultar las hojas de estilo a estos navegadores.

- Sí, recuerdo eso.

- En realidad, sólo con CSS1 no deberías tener demasiados problemas. Los tendrás sobre todo con Netscape Navigator 4. También con Internet Explorer 3, pero éste ya casi nadie lo usa. Es probable que el navegador aplique los estilos tan mal, que la página no se pueda usar o que simplemente quede horrible. Por ello es bastante normal usar el siguiente truco para evitar que estos navegadores lean una hoja de estilo:

```
<link rel="stylesheet" href="import.css" title="Hoja de Estilo" type="text/css" media="screen">
```

La hoja de estilo `import.css` sólo tiene una línea, que es parecida a esta:

```
@import url("principal.css");
```

donde `principal.css` es la auténtica hoja de estilo. Como NN4 e IE3 no entienden la regla `@import`, no importarán ninguna de las reglas contenidas en `principal.css`.

- Anda, qué ingenioso.

- Al hacer esto en estos navegadores tu página se mostrará muy sosa, en HTML puro sin colores ni adornos ni nada, pero por lo menos será perfectamente utilizable, que es lo importante.

- Sí, supongo que sí.

- De todos modos, aunque cada vez son menos los que usan estos navegadores, es posible crear una hoja de estilo sólo para

Netscape Navigator 4, y vincularla al documento de este modo, con un pequeño script de javascript:

```
   <script type="text/javascript">
<!--
if (document.layers) { // si está definido document.layers es NN4
    document.write(\'<link rel="stylesheet" href="nn4.css" type="text/css" media="screen">\')
}
// -->
</script>
```

Resulta que en NN4 las hojas de estilo sólo están activadas si también lo está el javascript, así que esto siempre funcionará. Si recibes muchos visitantes que usen NN4 puedes plantearte esta opción.

- Sí, pero de qué me vale, si no lo entiende bien...

- Bueno, algo sí que entiende, aunque mal. Puedes ver una lista de las propiedades de CSS1 soportadas por varios navegadores confeccionada por Eric A. Meyer. Con esta tabla te puedes hacer una idea de las propiedades que puedes usar, pero en todo caso lo mejor es que pruebes tu página con unos cuantos navegadores.

- Sí, claro, cómo si no tuviera otra cosa que hacer...

- Ya lo sé, ya lo sé. Pero bueno, recuerda que lo importante es el contenido.

- Sí, y pasárselo bien.

- Sí, eso también. Bueno, cuando hablemos de posicionamiento CSS tendremos que ver más trucos de estos porque aún hay muchas diferencias de implementación entre los navegadores que más se usan actualmente. Pero de eso hablaremos en su momento...

- Ya, un año de estos, supongo...

- De momento te puedes hacer una idea del tipo de malabarismos que hay que hacer si visitas esta página de Johannes Koch.

- Eemmm, bueno, la verdad es que no entiendo casi nada...

- Normal, pero no te preocupes. Bueno, creo que ya está bien por hoy...

- ¿Ya? Pero si acabamos de empezar...

- ¿Qué más quieres? Tienes que practicar con el FTP. Puedes bajarte el WS_FTP y seguir el tutorial que te he dicho, o intentarlo con otro programa. Algunos de los editores que vienen en la página sobre editores vienen con una función de FTP incluida (como el Stone\'s Web Writer o el HTML Kit por ejemplo). Ya sabes cómo asegurarte de que salgan tus acentos, por lo menos si no escribes cosas en idiomas raros. Y también sabes cómo vincular tus hojas de estilo para que tu página sea utilizable en navegadores viejos.

- Sí... Entonces, ¿de qué vamos a hablar la próxima vez?

- De posicionamiento CSS. Es algo complicado pero ya va siendo hora de meterse en eso. Después podremos hablar un poco de javascript y de HTML dinámico, y también de formularios.

- Bueno, bueno, paso a paso.

- Sí. Espero que me enseñes tu página cuando la hayas publicado. Si tienes algún problema, pregunta en la lista, ¿ok?

- Ok. Bueno, hasta la próxima, ¡y feliz año nuevo!

- Hum... gracias, igualmente. Adiós.

Tabla de contenido