Cómo hacer buenas páginas web

Continuamente veo páginas técnicamente mal diseñadas (ni siquiera las empresas gordas se salvan). Los culpables son webmasters que dicen saber HTML, pero que lo estudiaron hace años, cuando aún no existía CSS.

Esto intenta ser un "manual de buen comportamiento" para los que colaboran con Internet escribiendo páginas web. Explicaré cómo hacer que una página se vea bien en todos los navegadores, usando HTML 4.01 Strict, que luego te permitirá pasar a lenguajes mejores, como XHTML, sin esfuerzo.

No soy experto en esto, y lo que digo aquí es mi opinión, pero verás que coincide con la de cualquiera que entienda del tema. Por último, si necesitas ejemplos, puedes empezar analizando esta web (sólo la sección HTML), pues la considero bien hecha.

Septiembre 2004, Daniel Clemente Laboreo. Licencia FDL.

Índice

Ideas generales

Las características de una web bien construida son las siguientes:

Una web se tiene que poder ver en todos los navegadores

Y cuando digo en todos, me refiero a **todos**, no sólo los más comunes. El si se ve más bonita o más fea es otra historia, pero cualquier visitante debe poder entrar en cualquier sección sin problemas.

Esto es mucho más fácil de lo que parece, y es lo que quiero explicar en este artículo.

Separa el contenido del diseño

La regla de oro que hay que seguir al hacer una web. Todos los navegadores pueden

mostrar el texto de una página, pero algunos no soportan las imágenes, colores, animaciones, menús o efectos especiales. Si consigues hacer todo esto opcional (sólo para navegadores avanzados), te aseguras de que cualquiera pueda leer la página, que probablemente es lo que quieren los visitantes. Quien, además, quiera ver tu web bonita, ya se preocupará de usar un navegador avanzado.

La clave para separar el contenido del diseño es seguir estos pasos:

1. Escribir toda la estructura y cada sección de la web, sin preocuparse por el diseño. Esto consiste en dos pasos:
    1. escribir el texto (las palabras, frases, y todo lo que tengas que contar).
    2. usar el HTML para anotar la estructura del texto. Son etiquetas muy sencillas; sólo hay que ir revisando el texto e ir pensando "esto es un párrafo, esto otro, esto una cabecera, esto una lista, etc."
2. Una vez acabada la página (por ahora fea), se escribe un diseño en CSS ("hojas de estilo"), tal vez en un fichero aparte.

El paso 1 es para el contenido; el paso 2 para el diseño. Si alguien se pierde el diseño, se queda en el paso 1: con una web estructurada y con contenido.

Si luego quieres añadir más cosas a la web, ya no tienes que preocuparte del diseño. Y viceversa: si te cansas del diseño, puedes crear otro sin tener que tocar el código HTML ni las secciones que escribiste.

Más adelante explico qué es eso de HTML y CSS.

El código HTML tiene que ser válido

Para que todo funcione bien, tienes que usar el HTML que entienden los navegadores, no el que tú te inventes.

Podrías leerte las especificaciones del lenguaje HTML y compararlas con tu código, pero es más fácil que uses herramientas como el validador de HTML de la W3C (es quien creó el estándar HTML). Le pones la dirección de la página o un fichero de tu disco, y te dice qué errores le encuentra.

No hagas eso de probar en varios navegadores a ver si se ve bien; la prueba definitiva es ver si el código valida en el test o no. Si valida, todos los navegadores lo mostrarán bien, porque todos entienden HTML.

El código CSS también tiene que ser válido

CSS es otro lenguaje, y si no lo conoces puedes equivocarte al escribirlo. El validador de CSS de la W3C te dirá qué errores le ve a tu código, y si no le ve errores es que es correcto, porque también son ellos quienes se encargan del CSS.

Scripts en el servidor

Si la web es interactiva y tiene programitas que necesitan hacer cálculos, es mucho mejor si se ejecutan en el servidor, usando un lenguaje como PHP o mediante CGIs. Si usas JavaScript, applets Java, o cualquier otra cosa parecida, ya puedes estar seguro de que no todos los visitantes verán igual la web.

Lo de los scripts en el servidor es la única forma de asegurarse que reciban la misma web.

Accesibilidad

Se habla mucho de accesibilidad y usabilidad. Son temas muy extensos, pero fáciles de entender.

El contenido ha de poder llegar a todos

(Y el diseño, para el que quiera).

Ya sabes que la web se separa en contenido y diseño. Pues bien, una página es accesible si se puede llegar fácilmente a su contenido (aquello que la web dice). Por tanto, no tiene que estar escondido detrás de largos menús que a veces funcionan y otras no, ni tiene que depender del cómo ha configurado el usuario su navegador.

Como el contenido se escribe en HTML y todos los navegadores entienden HTML, con escribir buen HTML ya está casi todo hecho.

Respetemos a los ciegos (y miopes)

Siempre que se habla de accesibilidad se explica que los ciegos usan un "lector de pantalla", que es un programa que coge el texto de una web y lo lee en voz alta. Por supuesto, no explica las imágenes, y supongo que tampoco dice nada si hay colores o cambios en el tipo de letra.

Otros no son ciegos, pero necesitan hacer cambios en las páginas para poder verlas bien. Por ejemplo, pueden necesitar un contraste fuerte en los colores, o agrandar el tipo de letra. Entre éstos me incluyo yo, que a veces (después de muchas horas de ordenador) quiero quitarme las gafas y ponerlo todo más grande para que lo pueda leer mejor.

Al separar el texto del diseño, el usuario puede fácilmente descartar el diseño o incluso usar su diseño favorito, escrito en un fichero CSS, sin que entre en conflicto con el diseño original de la página.

He oído varios comentarios de desprecio del estilo de "¿Y a mí que me importan los ciegos? Si mi web no va dirigida a ellos... ¿Qué me pierdo si no entran los ciegos, un 0'5 por ciento?". Bueno, pues para vuestra información, Google es ciego. Cuando entra a tu web para añadirla a su base de datos, no puede ver imágenes, ni navegar por los menús JavaScript, ni usar el portal de entrada hecho en Flash. ¿En serio quieres marginar a los que no son como tú?

No es mi intención hablar sobre páginas especialmente orientadas a ciegos, pero encontrarás buenos consejos en este Llibre d'estil (en catalán) y en el validador Bobby para el WCAG 1.0, en inglés.

Estándares web

Lo de los estándares puede ser complicado en otros campos, pero por suerte en Internet

está todo muy claro:

¿Qué es un estándar?

Un estándar (o *standard*) es el modelo a seguir al hacer algo. Son documentos que dan los detalles técnicos y las reglas necesarias para que un producto o tecnología se use correctamente.

Un ejemplo típico es el de los códigos de barras, donde los etiquetadores y los fabricantes de aparatos lectores se entienden porque han seguido las mismas indicaciones sobre el cómo interpretarlos.

En la World Wide Web, algunos estándares actuales son: HTML, SVG, DOM, CSS, PNG, SOAP, XML, o HTTP. Lo siento, son todo siglas.

Es importante que entiendas que "estándar" no quiere decir "algo que es muy usado y común". Por ejemplo, no puedo decir que "el chino es el idioma estándar de la Tierra", ni tampoco el inglés. A este tipo de cosas se les llama "estándares *de facto*" o "pseudoestándares".

Por si no te habías dado cuenta aún, "estándar" es lo contrario de "tecnología propietaria", que es la que normalmente adopta una gran compañía y no pueden usarla los demás. Claro que también hay términos medios: tecnologías explicadas a medias, no explicadas pero conocidas, etc.

¿Quién los hace?

En el caso de Internet, la organización que hace los estándares es el World Wide Web Consortium (W3C), en la que todos pueden participar con sugerencias, críticas y mejoras.

Verás que el W3C no les llama "estándares" sino "recomendaciones", porque no se puede obligar a nadie a que los siga. Lo mismo pasa con otras organizaciones que hacen estándares, como ISO. O sea, que no es obligatorio seguir los estándares, pero si no lo haces te saldrán problemas.

¿Vale la pena seguirlos?

Naturalmente que sí; siempre es recomendable seguir las instrucciones del fabricante de un producto o tecnología.

Si lo haces todo tal como está explicado, saldrá bien. Y si sale mal, puedes echarle la culpa a otro: "Ahhh... yo he seguido las instrucciones que dais; si están mal no es mi problema". Y es que en muchos sitios no te darán soporte técnico si no sigues el procedimiento oficial.

Fíjate en si vale la pena: los programadores de navegadores web se han leído el estándar HTML de la W3C para saber en qué consiste el HTML. Si tú usas el mismo HTML (el que ha propuesto la W3C), todos te entenderán de la misma forma. Por eso una página con HTML correcto se ve bien en todos los navegadores.

Además, es más fácil y rápido escribir HTML correcto. Es mucho más sencillo de lo que

puedas pensar (si aprendiste el HTML hace tiempo, mira el código de esta web y verás).

Qué pasa si no se siguen

Imagina que hablas o escribes a un extranjero que conoce lo básico del español. Si empiezas a hablarle en tu propia jerga (con abreviaturas, faltas de ortografía, sin acentos o con expresiones inventadas), le costará entenderte. Tiene dos opciones:

- Intentar descubrir qué es lo que quieres decir con cada palabra.
- Directamente decirte que no te entiende, y que le hables claro.

De éstas, la primera opción no es nada cómoda para él, ya que tiene que pensar mucho e incluso usar su imaginación. Eso no te favorece, porque a lo mejor acaba creyendo cosas que tú no has dicho, sólo porque las ha entendido mal. Hasta puede pasar que le digas la misma cosa a dos extranjeros distintos, y cada uno entienda algo diferente (y eso que cualquier amigo tuyo te entendería...).

En cambio, si te avisa de que él conoce un idioma (español) pero que no es el mismo que el que tú le estás hablando, podrás rectificar y tener una conversación normal, cómoda para los dos.

Aunque no lo parezca, el estado actual de Internet es el primero: los navegadores han de imaginarse qué es lo que quería decir el autor de cada página web, porque, en efecto, el lenguaje que se habla (HTML) no es el que entienden los navegadores. Por eso es normal que cada navegador interprete de forma distinta la misma web.

Por suerte, el estándar XHTML (el que debe sustituir a HTML con el tiempo) se comporta como en el segundo caso: si tú haces un documento XHTML mal escrito, el navegador ha de pararse y decir que no sabe mostrarlo (porque es que realmente no sabe). Así, todos los documentos XHTML serán legibles y claros. Todos hablarán bien el lenguaje, y sólo hará falta evitar las "frases" sin sentido.

Algunas falacias (mentiras que la gente se cree)

Hay gente que habla de todo sin saber. Algunas frases que he oído -impactantes para cualquiera que conozca CSS y HTML- son:

"Los estándares limitan a los diseñadores web"

Alguien me ha llegado a decir: "¿Pero cómo voy a respetar los estándares, si sólo soy un diseñador, y además quiero usar Flash y JavaScript?".

Pues si te parece que ser correcto implica restringirse y no poder usar un montón de cosas, te equivocas. Si te propones aprender y usar los estándares (en contraposición a las tecnologías propietarias), tienes a tu disposición:

- HTML: lo que casi todos usan para hacer páginas web.
- CSS: si no fuera por él, no habríamos avanzado nada en la web. Lo más importante que un diseñador de webs debe saber.
- JavaScript: sí, JavaScript es un estándar.
- Flash, Java, y otros formatos: sí, insertar contenidos externos en una web forma parte del estándar HTML, aunque los plugins sean propietarios.

- XML: el futuro para estructurar datos; útil en cualquier sitio.
- PNG: excelente formato gráfico, mucho mejor que GIF.
- SVG: gráficos vectoriales para las páginas web.
- SOAP: para compartir información entre aplicaciones por HTTP.
- MathML: para poner fórmulas matemáticas.
- P3P: para controlar las políticas de privacidad de una web.
- Muchas más cosas: XHTML, DOM, HTTP, XSL, SMIL, DOM, y más.

Así que no digas que los estándares te limitan, sino que gracias a los estándares abiertos tenemos Internet.

"Los *estándares* de Internet Explorer son los más usados"

Que no... que los *estándares* los hace el W3C. Si te refieres a las tecnologías propietarias de Microsoft que están muy extendidas (tanto que parece "normal" tenerlas, como Windows), a eso se le llama "estándar de facto" o pseudoestándar.

Y no, las tecnologías propietarias de Microsoft (como VBScript, ActiveX, DHTML, o JScript) no son muy comunes, pero cuando se usan en un sitio web molestan **mucho**. Lo que sí que es común es que cada uno escriba el HTML como quiera, mezclando cosas de todos lados.

"Microsoft innova al salirse de los estándares"

Los estándares que cito arriba no son nada anticuados (como pueda pasar con el diccionario de la RAE); al contrario: cada semana salen nuevos documentos, propuestas e informes técnicos, y se seguirá trabajando en ellos para simplificarlos y hacer todo más sencillo. Lo que se puede hacer ahora con CSS, PNG, o SVG es maravilloso.

En cambio, cuando Microsoft, en uno de sus productos -que siempre son anticuados- implementa mal un estándar por voluntad propia, crea el caos porque a partir de entonces habrá dos formas de hacer la misma cosa: la forma buena, y la mala. Muchos usarán la mala, y como algunos productos de Microsoft son muy usados, el mundo se llenará de cosas mal hechas. Eso es lo que pasa con las páginas web.

Resumiendo: los estándares no son anticuados. Niguna empresa tiene que renovarlos; quien piense que son malos que participe en su creación. Microsoft estropea todo al no implementarlos de la forma correcta.

Alguien decía una vez: "voy a empezar a conducir en dirección contraria, ¿es malo? No, porque así obligo a los demás a ir más atentos y habrá menos accidentes, ¿no?"

Navegadores

A todo lo que accede a la WWW se le llama *User Agent* (agente usuario). Los UA más comunes son los navegadores, pero otros UA son los robots de los buscadores, los teléfonos móviles, o los asquerosos programas de los *spammers* buscando direcciones de e-mail.

No sólo hay dos; hay cientos

Seguro que has oído hablar de Netscape y de Internet Explorer. Pues en realidad hay tantos navegadores que no podrías contarlos todos, y esos dos (los más conocidos) son precisamente lo peor que ha visto Internet (sobre todo, IE). Te puede interesar mi artículo sobre alternativas a Internet Explorer para Windows.

Ninguno implementa al 100% los estándares

Cada navegador se comporta de una manera distinta (incluso cuando usan el mismo motor), así que no esperes conocerlos todos. Lo que vas a tener que hacer es que tu página se vea bien en cualquiera de ellos, usando los lenguajes estándares.

Desgraciadamente, no hay ningún navegador que soporte **todos** los estándares de la W3C a la vez, porque son muchísimos y muy extensos. Por suerte, el HTML (versión 4.01, por ejemplo) sí que lo entienden todos, y casi sin errores, pues es un requisito indispensable y además es sencillo.

Por eso una página que usa HTML correcto es accesible desde cualquier navegador. Los demás estándares (CSS, SVG, JavaScript, PNG, ...) sirven para complementar la página, y no siempre se verán en todos lados. Es inevitable.

Lo peor viene cuando un navegador **dice** soportar los estándares, pero no los implementa bien. Eso hace el Internet Explorer con algunas características de CSS o con los PNGs con canales de transparencias *alpha*. Por eso siempre se dice que Internet Explorer es el peor en este tema, y alguien que pruebe páginas web no debería usarlo mientras las programa.

Como navegadores que sigan fielmente los estándares se suelen poner de ejemplo el Amaya (del W3C) o Mozilla (o Mozilla Firefox), que soporta muchos y lo hace bien. Hay que destacar que todos los navegadores que no son IE hacen bien su trabajo: puede que hagan muchas cosas o pocas, pero todo lo que hacen lo hacen bien. No suelen dar quebraderos de cabeza: una cosa, o funciona bien o no funciona.

Nunca hagas una página específica para un navegador

Es muy fácil hacer una página que se vea bien en todos los navegadores, así que no te metas en el tema de hacer diferentes versiones de la misma página, una para cada navegador "soportado". Nunca podrás "soportarlos" todos, y lleva mucho trabajo (inútil).

Es absurdo ver mensajes que digan "Página optimizada para..." y ahí un navegador o resolución de pantalla (¿vas a tener que cambiar la resolución o el navegador sólo para ver una página?). Tus webs tienen que ser "Optimizadas para verse con cualquier navegador".

Más falacias (mentiras que la gente se cree)

Encontrarás muchas más, desmentidas, en el popup que muestro a los usuarios de Internet Explorer, que escribí hace unos meses.

"No hay ninguna página que no pueda ver bien con IE"

Internet Explorer muestra mal algunas páginas; las destroza a su manera. Por eso, muchas veces no estás viendo **bien** la página. Si la quieres ver tal como ha escrito el

diseñador, usa un navegador que siga bien los estándares. Entonces verás el aspecto real de cada web.

Lo lógico sería que el "aspecto real" y "lo que se ve en IE" coincidieran, pero no siempre es así.

"Es mejor diseñar una web para IE porque es lo más usado"

No, es mejor diseñar una web visible desde cualquier navegador. Encima más fácil y no da problemas.

Además, mientras haces la web, no es nada recomendable usar IE "para probar si queda bien", porque la modificarás hasta que se vea bien en IE, y eso puede requerir salirse del estándar, haciendo que se vea mal en el resto de navegadores (que no tienen la culpa).

Sobre HTML

Si eres de los que usan habitualmente cosas como `<font>`, `<br>`, `<b>` o `<table>`, lee esto, porque te ayudará. El HTML correcto es más sencillo que el que tú conoces, y permite hacer muchas más cosas.

¿Qué es el HTML?

Si las páginas fueran sólo líneas y líneas de texto, serían muy aburridas. Para evitar esto, el autor puede dejar escrita la estructura del texto para que los usuarios puedan -mediante sus navegadores- dar un formato especial a algunas secciones.

HTML es sólo para describir la estructura de una web. El autor le cuenta al navegador cómo es la web; y es trabajo del navegador decidir cómo va a mostrar cada sección (colores, tamaños, tipos de letra, ...). Si el autor quiere además especificar el diseño, puede usar CSS (hojas de estilo), que explico más adelante. No sirve el HTML.

¿Es necesario usar HTML?

No, hay más formas de estructurar contenido, por ejemplo XML o XHTML (que es una mezcla de XML y HTML).

Con el tiempo dejará de usarse HTML en favor de XHTML, que cumple mejor con su objetivo.

¿Es necesario saber HTML para hacer una web?

No. Hay programas con los que puedes dar la estructura al texto cómodamente, y generan código HTML correcto (excepto Frontpage). Yo uso el Mozilla Composer para ahorrar trabajo, aunque luego lo reviso a mano.

De todas formas, va muy bien saberlo, y es un lenguaje muy simple.

¿Es fácil escribir HTML?

Sí, tanto que hay muchos cursillos para principiantes en cualquier academia de informática. No hay que saber programar, y sólo hace falta un editor de textos.

Esta "facilidad" hace que mucha gente consiga hacer páginas casi sin saber qué es lo que ha escrito, con código de baja calidad y complicándose demasiado.

¿Es fácil escribir HTML correcto?

Para un principiante y a mano, cuesta, sobre todo cuando no sabes qué es correcto y qué no.

No obstante, se hace muy sencillo si tú evitas complicarlo. Hay muy poco código HTML en una página bien hecha; mira ésta como ejemplo. Eso es lo que explicaré en la sección de Consejos.

También hay un programa, HTML Tidy, que puede buscar errores sencillos en tu código y corregirlos un poco. El código resultante es muy complicado de entender; es mucho mejor si lo escribes bien desde el principio.

Cursillo de HTML

En inglés: puedes consultar la propia especificación HTML 4.01 del W3C. Es lo mejor que hay, porque no tiene fallos. Si hicieron algún error al escribir, ya no es un error, ¡ahora es oficial!.

En español: como siempre, no he encontrado ninguno que supere mi "test de calidad": la mayoría de cursos enseñan HTML no correcto. Algunos medianamente buenos son: SelfHTML, el de Daniel Rodríguez, LuCAS.

Si lo que te interesa es el XHTML, tienes un tutorial de XHTML muy bueno hecho por Belén Albeza (BenKo). Consulta su web para más información.

Cómo es una dirección (URI)

Una URI (eso incluye a las URL) tiene el nombre del protocolo (suele ser `http`), el `://` seguido del nombre de la máquina, y luego la ruta del archivo al que se accede. Por ejemplo,

```
http://www.danielclemente.com/html/index.html
http://www.danielclemente.com/html/
mailto://142857@ozu.quitando-esto.es
news://netscape.public.mozilla.svg/
```

son URIs. Fíjate en que la primera hace referencia a un arhivo de mi servidor, y la segunda a un directorio, por eso lleva la barra al final. Si no pusieras la barra, se te redirigiría a la dirección correcta; pruébalo: http://www.danielclemente.com/html (sin la barra final).

Extensión de los archivos: ¿ `html` o `htm` ?

¿Por qué `index.html` y no `index.htm`, si también funciona y es más corto?

Bueno, me gusta llamar a las cosas por su nombre, y esto es una página HTML. HTM no quiere decir nada; el lenguaje siempre se ha llamado HTML.

La culpa de este lío, es, como siempre, de Microsoft. Desde que compraron QDOS (MS-DOS), ya limitaban la extensión de un archivo a 3 caracteres. Tardaron 14 años en copiar el modelo de UNIX y Linux, en el que el nombre y la extensión pueden ser tan largos o cortos como se quiera.

Así que: llama a las cosas por su nombre: `.xhtml` para el XHTML, `.html` para HTML, `.php` para PHP, `.xml` para XML, etc. ¿Por qué habríamos de llamar XM al XML?

Efectos HTML

He visto que muchos entran en mi web buscando en Google "efectos html". Pues que quede claro:

HTML es un lenguaje para estructurar texto.

No es para hacer animaciones, ni efectos especiales, ni juegos. Probablemente te interesa aprender CSS para hacer virguerías inimaginables con el diseño; eso sí que es posible.

Versiones de HTML

Esto es importante. Mucha gente se lo olvida y les da problemas de todo tipo (cada navegador les muestra la página como quiere).

¿Cómo?, ¿que hay versiones?

Claro que hay versiones. Se hizo estándar en 1995, y desde entonces las cosas han cambiado (aunque el lenguaje no se ha modificado mucho).

En cada página que hagas, tendrás que decirle al navegador qué versión de HTML estás usando (no puede detectarlo). Eso se hace con la primera línea, la del `!DOCTYPE`. Luego lo explico.

HTML

Los estándares HTML son el HTML 2.0, el HTML 3.2, el HTML 4.0 y el HTML 4.01. La verdad, no creo que se hagan más, porque el HTML ya está anticuado (comparado con XHTML).

Todos son comprendidos por los navegadores actuales, pero el que debes usar es el HTML 4.01. Aún se puede usar sin problemas el HTML 3.2, pero le falta alguna etiqueta interesante ya que en esos tiempos es cuando empezaron a introducir la filosofía CSS.

Además, cada versión tiene variantes. Tienes que elegir una. Por ejemplo, el HTML 4.01 tiene tres:

- HTML 4.01 Strict: el normal. Puedes usar las etiquetas de HTML 4.01 y ya está.
- HTML 4.01 Transitional: una mezcla de todos los HTML, en la que se aceptan las etiquetas obsoletas. Se llama Transitional porque está pensado para los que no se

atreven a usar el Strict, pero les gustaría en el futuro. En mi opinión no va bien usarlo, es un lío tener tantas formas distintas de hacer la misma cosa (unas buenas y otras malas).
- HTML 4.01 Frameset: lo del Transitional, y soporte para frames. Anticuado, como los frames.

Yo explicaré y usaré HTML 4.01 Strict, aunque XHTML me parece mucho mejor. Pero como es muy fácil pasar de HTML 4.01 Strict a XHTML 1.0 (casi no hay cambios), prefiero explicar el HTML. Aparte... es que no podría explicar muchas cosas sobre "XHTML correcto", porque es muy fácil escribir XHTML correcto (sigue leyendo).

XHTML

HTML está anticuado. Sus normas son muy poco estrictas, y eso da muchos problemas: mucha gente escribe el código "a medias" (por ejemplo, abre etiquetas y no las cierra), y dejan que sea el navegador el que "arregle" la página. Por eso un navegador web es muy complicado y suele tener problemas de compatibilidad.

XHTML trae las ventajas del XML, que son muchas. Las conocidas son que los nombres de etiquetas van en minúsculas, que hay que cerrar todas las que se abren (y sin anidarse), y que los atributos van entre comillas.

Pero yo quiero destacar otras dos ventajas muy importantes del XHTML:

- Si un programa o navegador que está analizando el XHTML encuentra XML mal formado (erróneo), **está obligado** a parar y decir que es incorrecto. Eso simplifica enormemente el código de un navegador web, y hace que en Internet sólo pueda existir el código XHTML bien escrito, compatible con todos los analizadores de XML. (Aunque eso no asegura que las etiquetas estén bien usadas).
- Los navegadores XHTML viejos no fallarán cuando se use un XHTML lleno de etiquetas nuevas, porque en XML las etiquetas desconocidas o que no hacen falta se ignoran junto con su contenido. No hay que hacer trucos como lo de comentar el código JavaScript para que no se muestre en navegadores HTML antiguos.

XHTML tiene que sustituir a HTML en los próximos años, y ya lo está haciendo. Las páginas tienen extensión `.xhtml` y el servidor web tiene que estar configurado para servirlas como `application/xhtml+xml` (no como `text/html`).

XSL

XSL permite que el navegador (o el servidor) transforme los documentos XML y haga cosas como ordenar una tabla, o hacer búsquedas, o bucles, cosas que no pueden hacerse con HTML ni XHTML. Lo malo es que necesita que el navegador lo soporte, pero muchos navegadores ya lo hacen.

XSL consta de los estándares XSLT, XPath y XSL-FO.

¿Cuál elijo, y cómo?

Si ya sabías HTML pero nunca has hecho todo eso de separar contenido y diseño, o todos estos estándares te suenan a chino, usa el HTML normal, en concreto la versión 4.01, que es la última. Elige la rama Strict directamente si te gusta el tema, o el 4.01

Transitional si te cuesta entenderlo.

El XHTML es para los que entienden qué es lo del XML, tienen otros programas que trabajan con datos de la web, o simplemente les gustan mucho los estándares.

Para decir qué lenguaje se utiliza, hay que poner una línea al principio de la página. No es una etiqueta, por tanto es algo rara y no hay que cerrarla ni ponerla en minúsculas. Eso sí, hay que ponerla.

Para HTML 4.01 Strict (recomendado):

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

Para HTML 4.01 Transitional:

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

Para XHTML: tienes las declaraciones necesarias en la especificación XHTML del W3C.

CSS

Es el requisito de cualquier diseñador de webs (aunque no sepa HTML).

Para qué sirve

Hasta ahora he explicado que el HTML sirve para estructurar el texto: ya tenemos la página dividida en secciones: párrafos, enlaces, cabeceras, citas, imágenes, etc.

Queda aplicarles un estilo. Eso consiste en decir, por ejemplo, que "las cabeceras van en rojo y subrayadas, las imágenes tienen 2 cm. de margen y un borde de 1 píxel, la primera línea de cada párrafo está indentada, el interlineado es de 1'5, y la primera fila de cada tabla tiene el fondo azul". Todo eso se puede hacer con CSS sin necesidad de tocar el HTML.

En general, el CSS sirve para aplicar un estilo a todos los elementos del mismo tipo a la vez. Bueno, permite hacer muchas más cosas, pero lo mejor para descubrirlo es verlo en acción.

Cómo puede combinarse con el HTML

Lo más normal es escribir todo el código CSS en un fichero externo, de extensión `css`, y luego incluir en el `<head>` de cada página el código:

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

Esto permite usar el mismo estilo para varias páginas HTML distintas. No sólo eso: haciendo esto te llevas por completo el diseño a otro archivo, de forma los navegadores sencillos, que no soportan hojas de estilo, no tendrán que cargar código innecesario (por eso van tan rápido).

Si prefieres dejar el código CSS en la misma página (sin usar archivos externos), se hace

desde dentro del `<head>` con la etiqueta `<style>`:

```
<style type="text/css">
/* Aquí va el código CSS */
/*   Recuerda que esto va dentro del <head> */
</style>
```

También se puede definir un estilo para un solo elemento, escribiendo el código dentro de su atributo `style`, aunque no lo recomiendo porque complica el código. Ah, y también se pueden incluir archivos CSS externos con la orden `@import` de CSS, pero eso ya es más complicado.

Si definieras un estilo usando todas las formas a la vez, la prioridad que se seguiría es:

- Atributo `style` del elemento (estilo específico para un solo elemento)
- Estilo definido en la cabecera con la etiqueta `<style>`
- Estilo definido en un archivo externo, y asociado a la página con `<link>`

Formato

Un ejemplo de código CSS es:

```
h1 { color: red; text-decoration: underline; }
p { text-indent: 20px; border: 1px dotted gray; line-height: 200%; }
```

Esto define estilos para lo que va entre etiquetas `<h1> </h1>` (que son cabeceras), y lo que va entre etiquetas `<p> </p>` (que son párrafos). Hace que las cabeceras salgan en rojo y subrayadas, y los párrafos con interlineado doble, con la primera línea un poquito más para la derecha, y con un borde gris y a puntitos alrededor de cada párrafo.

El formato general es:

```
etiqueta {propiedad1:valor; propiedad2:valor; propiedadn:valor;}
```

y así para cada etiqueta a la que quieras dar estilo.

Dónde aprender CSS

Hay muchos tutoriales de CSS en todos los idiomas, pero en mi opinión, las mejores formas de aprender son:

- Mirando el código de webmasters que sepan más que tú. Cada vez que veas algo un poco bonito, mira el código para ver cómo está hecho. No importa si la página está en inglés o en tailandés; el CSS es el mismo en todos lados.
- Consultando la especificación CSS del W3C cada vez que haya cosas que no entiendas. Así aprenderás a entender y perfeccionar lo que ya sabías.

Sitios donde puedes ver la potencia de CSS y aprender cosas nuevas: Zen Garden, sea mus n squirrel, Position is everything, y muchos blogs como el de minid o Toad (que escribió un tutorial de CSS muy bueno). Recuerda que necesitarás un navegador moderno para ver las páginas tal como ha querido el autor (no vale Internet Explorer).

También puedes empezar mirando el código de esta página, que no es muy complicado.

Varios ejemplos

Una muestra muy variada de cosas que se pueden hacer con CSS:

```
/* Las cabeceras h1, en un color azulado y en mayúsculas */
h1 {color: #0077ff; text-transform: uppercase;}

/* El p que está metido dentro de un div, con imagen de fondo */
div p {background: url(fondo.png);}

/* El li que es hijo directo del ol, con borde azul */
ol > li {border: 1px solid blue;}

/* Los enlaces, un poco más grandes al pasar el ratón */
a:hover {font-size: 120%;}

/* Párrafos y listas, con indentación */
p, li {text-indent: 15px;}

/* Justo después de una imagen, no indentar */
img + p {text-indent: 0;}

/* Las imágenes con class="separa", con un poco de margen */
img.separa {margin: 20px;}

/* El bloque div con id="tabla" tiene el fondo rojo, la letra blanca,
  ocupa el 75% de la pantalla (centrado), y tiene un margen interno */
div#tabla {background: red; color: white;
           width: 75%; margin: 0 auto; padding: 20px;}
```

Cambiar el color de las barras de desplazamiento

Mucha gente entra a mi web buscando "cómo cambiar el color de las barras de desplazamiento" con HTML o CSS. Pues **no** se puede, al menos con las versiones de CSS que hay al escribir esto (2004).

Como mi navegador soporta temas y personalización, las barras de scroll las modifico yo a mi gusto, no las páginas. Una página controla el trozo de ventana en el que se muestra, no los iconitos, la disposición de los botones, o los menús. Todos estos elementos de navegación han de ser reconocibles por el usuario.

Hace tiempo escribí el cómo no debe hacerse, en este documento: cambiar el color de las barras de desplazamiento **sin** CSS. Que quede claro que no es CSS, sino algo que ha inventado únicamente Microsoft sin pedir consejo a nadie, y le ha llamado con el mismo nombre.

Consejos HTML

Ahora llega lo interesante. Pongo errores típicos de webs, que veo muy a menudo (también en páginas que hice yo hace tiempo), y cómo corregir las malas costumbres para al final conseguir hacer una web accesible por cualquier navegador, con diseño y

contenido por separado, y que sea fácil de mantener.

Son trucos sueltos y desordenados, y seguro que voy añadiendo más con el tiempo.

La primera línea debe ser el DOCTYPE

Es obligatorio ponerla, y muy importante. Es para decirle al navegador qué versión de HTML es la que usas en la página. Si no se pone, el navegador no tiene forma de detectarlo, así que entra en modo *Quirks* (modo chapuzas) e intentará arreglar a su manera las cosas que no entienda, imáginandose qué quiere decir cada etiqueta, y representándolas tanto si son válidas como si no.

Si se usa una versión de HTML que sea Strict, entonces el navegador usa el modo *Standards compliance* (cumplimiento de los estándares), y cada etiqueta tendrá el significado normal.

Por ejemplo, en esta web la primera línea es:

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

Recuerda que no es una etiqueta, y por eso es tan rara, va en mayúsculas, y no se cierra.

Estructura básica de una web

Un ejemplo de lo mínimo que tiene que tener:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <title>Mi web. Qué bonita que es mi web.</title>
</head>
<body>
Aquí va el contenido.
</body>
</html>
```

El título, lo más importante

Dicen que el `<title>` es, con mucha diferencia, lo más importante de una web.

La verdad es que sí que es importante: lo usan los buscadores para indexar la página, pero (aún más importante) lo usamos las personas para saber en qué web entrar cuando un buscador muestra cientos de ellas. Así que elige un buen título, y ni se te ocurra dejarlo en blanco.

Escribe todas las etiquetas en minúsculas

Se aceptan tanto mayúsculas como minúsculas, pero es mejor que elijas un estilo fijo y no lo vayas cambiando. Yo he decidido las minúsculas por muchas cosas:

- es más fácil de leer.
- es más fácil de escribir.
- en XML (y XHTML) sólo se aceptan minúsculas, así que el cambio a XHTML será muy fácil.
- una página se comprime mejor cuando se usan etiquetas en minúsculas. Bueno, vale, esto es una razón algo freaky...
- CSS va en minúsculas también.

Cierra todo lo que abras (con excepciones), y en orden

No está permitido hacer cosas como `<td><a>enlace</td>`, ni `<ul><li>uno<li>dos<li>tres</ul>`, ni `<strong>código <em>incorrecto</strong></em>`. Eso causa desastres en muchos navegadores: imagina que empiezas un enlace al principio de la página, y no lo cierras nunca. Entonces ¡toda la página forma parte del enlace! y puede salir toda subrayada (es muy asqueroso).

Hay algunas etiquetas que no se cierran porque no tienen contenido por dentro, como `<img>` o `<br>`. En XHTML es obligatorio cerrarlas, y para no tener que escribir `<br></br>`, se permite `<br />`. Pero eso es XHTML... de momento en el HTML no cierres `<br>` y ya está.

Los atributos, siempre entrecomillados

En realidad sólo es obligatorio poner entre comillas los atributos que tienen caracteres no alfanuméricos, por ejemplo en `<body bgcolor="#E099F5">` (por el `#`).

Pero es mucho mejor que las pongas siempre, porque así no hay que pensar cuándo sí y cuándo no, y además es como hay que hacerlo en XML y XHTML. De todas formas, no hace falta poner casi atributos en el HTML correcto.

Puedes usar varios tipos de comillas, pero lo más normal es la comilla doble, `"`.

Especifica el juego de caracteres (charset)

Tu servidor ya tiene que estar preparado para mandar el tipo MIME del documento (que debe ser `text/html`) y la codificación (que es ISO-8859-1 para el español). Pero, por si acaso al navegador no le llega esta información, puedes decírsela con esta etiqueta dentro del `<head>`:

```
<meta content="text/html; charset=ISO-8859-1" http-equiv="content-type">
```

Así también se le dice el juego de caracteres a usar, y te aseguras de que los acentos se verán bien. En algunas páginas usan UTF-8 (Unicode) como *character encoding*, pero esto sólo te hará falta si usas caracteres raros como consonantes acentuadas o letras tibetanas o japonesas. Si dices usar Unicode y es mentira, los navegadores se confundirán, porque algunos caracteres de Unicode ocupan más de un byte.

No uses `<font>`

...para nada. No es necesaria en absoluto. Con CSS se puede hacer mucho mejor, eligiendo los colores/fuente/tamaño una sola vez, y aplicándolos a todos los elementos a la vez.

Si quieres dar un estilo especial sólo a unas letras o palabras, mira el siguiente consejo:

Dar formato a un trozo de texto o un trozo de página: `<span>` y `<div>`

Estas etiquetas las vas a usar mucho. Por sí solas no hacen nada; tienes que darles estilo.

Primero, una explicación importante sobre el cómo funciona el HTML:

Cada elemento HTML es (básicamente) de tipo *inline* o *block*. Los elementos de tipo *inline* siguen el flujo del texto, y se pueden meter como si fueran una palabra más. Por ejemplo, las imágenes y los enlaces se comportan así, porque puedo poner un enlace como si fueran palabras normales (le he puesto un borde para que veas en qué consiste el elemento).

En cambio, un elemento de tipo *block* rompe el flujo de texto: hace que se le dedique una línea nueva y toda la anchura de la página. Un ejemplo de esto son las cabeceras (`<h1>`, etc): no pueden compartir espacio con otro elemento. Los párrafos también son así: fíjate qué pasa si pongo

un párrafo

en medio de una línea.

Bueno, pues el `<span>` es *inline* y el `<div>` es *block*. Cuando quieras dar formato a unas cuantas palabras, sin romper el flujo del texto, mete esas palabras dentro del `<span>`, y cuando quieras formatear un montón de elementos juntos (un trozo de página, de una línea o varias), mételo todo dentro de un `<div>`.

Un ejemplo (y el cómo queda):

```
<p>Puedes aplicar <span style="font-size: 120%; text-transform: uppercase; color:
blue; cursor:pointer;">muchos efectos</span> a un trozo de texto sin necesidad de
usar la etiqueta font.</p>
<div style="text-align: center; border: 3px dashed orange; background: yellow;
font-style:italic;">
<p>Y puedes formatear todo un trozo de página a la vez...</p>
<p>si lo metes dentro de un div.</p>
</div>
```

Puedes aplicar MUCHOS EFECTOS a un trozo de texto sin necesidad de usar la etiqueta font.

Y puedes formatear todo un trozo de página a la vez...

si lo metes dentro de un div.

Recuerda que no es cómodo usar el atributo `style`. Es mejor definir clases (atributo `class`) y luego darles estilo a todos los de la misma clase a la vez. Lo explico en otra sección.

Evita el `<br>`, usa párrafos: `<p>`

El contenido de casi cualquier web se organiza por párrafos (mira ésta, por ejemplo). Los párrafos no hay que separarlos por saltos de línea (`<br>`), sino que sólo con poner `<p>` al principio y `</p>` al final ya quedan bien ordenados y con la separación adecuada.

Muy pocas veces hay que escribir un salto de línea de verdad... sólo se me ocurre el caso de que quieras poner un listado del código fuente de un programa: consiste en varias líneas (incluso algunas en blanco), una debajo de otra. Cada línea no es un párrafo por sí sola, así que sería más inteligente bajar a la siguiente con un `<br>`. Pero en general no hace falta; es mucho más cómodo usar párrafos.

No hace falta usar ` ` para hacer márgenes

Para hacer que la primera línea de cada párrafo quede un poquito más a la derecha, muchos ponen tres espacios: `   `. Eso es una chapuza; es mejor usar párrafos y aplicarles la siguiente hoja de estilos:

```
p {text-indent: 20px}
```

Con esto, **todos** los párrafos tendrán la primera línea 20 píxels a la derecha del margen. Elige bien las unidades; ponlas en cm. o mm. si te aclaras más, aunque quizás iría mejor no usar unidades absolutas sino en relación al tipo de letra actual.

Entonces, ¿cuándo hay que usar ` `? Pues sólo cuando quieras dibujar un espacio en blanco. Cuando haces un diseño, no creo que pienses "aquí me iría bien poner cinco espacios...", sino "esto lo quiero un poco más a la derecha", y para eso se usa el CSS. Si alguna vez realmente necesitas poner los cinco espacios (ni 4 ni 6), pon ` ` cinco veces.

`<center>` no existe

La etiqueta `<center>` y el atributo `align` (como en `<table align="center">`) no dejan muy claro su funcionamiento (¿deben alinearse ellos dentro de la página, o centrar el texto que viene a continuación?). Con CSS se puede especificar mejor los temas de centrado, pero hay que entender un poco qué es lo que queremos centrar, y recordar lo del tipo de los elementos (*block* o *inline*).

Para centrar el contenido que hay dentro de un elemento (por ejemplo, todo el texto de dentro de un `<div>`), basta con el estilo `text-align:center;`. Por ejemplo:

```
<div style="border:1px solid green; background:rgb(50,240,60); width:60%;
text-align:center;">
Este texto está centrado.
</div>
```

Y queda:

Este texto está centrado.

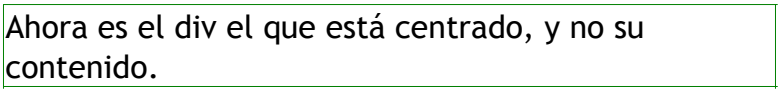
En cambio, si lo que quieres centrar es el propio `<div>` (u otro elemento tipo *block*), y no su contenido, tienes que darle una anchura (con `width:75%;`), y decir que el

navegador se ocupe automáticamente de los márgenes izquierdo y derecho. Se hace con `margin-left:auto; margin-right:auto;`, o, abreviado: `margin: 0 auto;`. Las dos formas hacen lo mismo, pero si te interesa entender por qué la corta funciona, busca información sobre "CSS *shorthand*".

Ejemplo para alinear un bloque:

```
<div style="border:1px solid green; background:rgb(50,240,60); width:60%; margin:
0 auto;">
Ahora es el div el que está centrado, y no su contenido.
</div>
```

Y queda:



Así ya sabes centrar tanto el contenido de un elemento, como cualquier elemento de tipo *block*. ¿Cómo se centraría un elemento *inline*, como una imagen o un `<span>`? Pues podrías meterla dentro de un `<div>` y usar `text-align:center; margin: 0 auto;` combinando los dos métodos anteriores, pero es más fácil pedirle que se comporte como un bloque, con `display:block; margin: 0 auto;` (así te ahorras el `<div>`).

El atributo `align` no existe

Lee el consejo anterior para ver cómo centrar cosas.

He visto que mucha gente usaba el `align="center"` en los `<td>`, para centrar el contenido de las celdas. Es mucho más fácil con CSS:

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

¡Y ya están todas las celdas de todas las tablas centradas! Si te interesa escoger sólo algunas celdas, estudia lo de darles nombre y clase (atributos `id` o `class`, que explico más adelante).

`<nobr>` no es necesario

Algunos navegadores hacen que el texto entre un `<nobr>` y `</nobr>` no se separe en varias líneas si la ventana en la que se muestra es pequeña.

En realidad, cuando usas esta etiqueta es porque quieres dibujar un espacio en blanco (quieres que sea como una letra más de una palabra, y asegurarte de que se dibuja en pantalla como las demás), y lo correcto sería usar el ` ` y olvidarse del `<nobr>`.

Lo bueno es que se puede hacer más fácil por CSS:

```
table tbody tr th {white-space:nowrap;}
```

Así, todas las cabeceras de las tablas ocuparán siempre una sola línea.

No uses `<i>`, `<b>`, `<u>`, sino `<em>`, `<strong>` y CSS.

Cuando estás escribiendo algo y llegas a una palabra especial, supongo que no pensarás "ay, esto tiene que ir cursiva" sino en "quiero resaltar esto". Lo de ponerlo en negrita, cursiva o subrayado forma parte del diseño, y ya hemos quedado en que el diseño se hará al final, cuando esté todo escrito.

De momento, lo que tienes que hacer es marcar las palabras a las que quieres dar énfasis con la etiqueta `<em>`, y si quieres dar un énfasis aún mayor, `<strong>`. Lo normal es que los navegadores representen el *énfasis* en cursiva y el **énfasis mayor** en negrita, pero lo bueno es que puedes cambiarlo mediante el CSS (incluso podrías desactivarlo).

Cuando realmente quieras especificar un estilo concreto, puedes usar `font-weight:bold` para **negrita**, `font-style:italic` para *cursiva*, `text-decoration:underline` para subrayado, y muchas más cosas, como `text-decoration:overline` para una línea por encima, o `text-decoration:line-through` para ~~tachado~~.

Cuidado con los & en las direcciones (URI)

Para poner caracteres especiales (como acentos) hay que escribir cosas como `&aacute;` (se llaman "entidades"). Empiezan por `&` (se llama "ampersand") y acaban por `;`. Por eso, cada vez que el navegador encuentra un `&`, se cree que vas a escribir una entidad, y ésa es la razón por la que hay que poner `&amp;` para dibujar un ampersand de verdad en la página.

Pues las URIs (y URLs) no son una excepción: si una dirección cualquiera (de una imagen, enlace, o lo que sea) tiene ampersands en su interior, los debes escribir como `&amp;`. Por ejemplo, si quieres poner este enlace en tu web: `http://www.google.es/search?hl=es&q=142857` , tienes que hacer esto:

```
<a href = "http://www.google.es/search?hl=es&amp;q=142857"> Informaci&oacute;n
sobre el 142857</a>
```

Tranquilo, que ni siquiera los navegadores viejos o simples se van a confundir. Precisamente esto les ayuda a no liarse con las *entities*.

Aprovecha las listas

Las listas (sobre todo `<ul>` y `<ol>`) son los elementos con los más se puede jugar al aprender CSS. Sólo con HTML son muy aburridas, pero añadiendo pocos estilos se pueden hacer maravillas como menús de navegación, botones, pestañas, tablas, y muchas más cosas.

En las páginas web abundan muchas listas, aunque no te hayas dado cuenta. Son una forma muy buena de estructurar la información, aparte de ser sencillas y de verse bien en todos los navegadores.

Puedes mirar páginas como Listamatic para probar las posibilidades de las listas.

Identificar elementos: `class` e `id`

Muchas veces querrás aplicar un estilo a varios elementos, pero no a todos a la vez; por ejemplo, si en tu página tuvieras muchos `<div>` con ayuda para los visitantes, y quieres que salgan todos igual, con el mismo estilo, no te iría bien una hoja de estilos como:

```
div {border:2px solid black; background:yellow;}
```

ya que cambiaría el diseño de todos los `<div>` de la página (aunque no sean los de ayuda). Hace falta algo para elegir unos pocos elementos concretos, y aplicar el estilo sólo a esos.

Como siempre, hay una solución elegante y sencilla que no requiere copiar y pegar estilos. La solución es distinta dependiendo de si hay varios elementos a los que aplicar el estilo, o sólo uno.

Cuando tenemos varios elementos parecidos que queremos agrupar, lo que hay que hacer es inventarse un nombre (nombre de clase) para asignárselo a todos ellos (porque pertenecerán a la misma clase). Eso se hace con el atributo `class`. Por ejemplo, en todos los `<div>` de ayuda tendremos que poner `<div class="ayuda">` (en los que no son de ayuda, `<div>` sólo).

Entonces les podemos cambiar el estilo a todos los de la clase a la vez, con el CSS. Se hace así:

```
div.ayuda {border:2px solid black; background:yellow;}
```

Eso quiere decir: todos los `<div>` que pertenecen a la clase `ayuda`. También podrías poner `.ayuda` para referirte a cualquier elemento que pertenezca a la clase `ayuda` (sea `<div>` o no), pero la primera forma es más clara.

En cambio, habrá veces en las que el elemento al que quieres dar estilo es único, o sea, sólo sale una vez en la página. Lo puedes hacer con `class` igual que antes, claro, pero si prefieres dejar constancia de que el elemento es único, dale un identificador con el atributo `id`; por ejemplo `<div id="titulo">`. No puede haber `id`s repetidos ya que son únicos.

Luego, en la hoja de estilos, se pone:

```
div#titulo {border:12px solid blue; background:gray; color:black;}
```

con un `#` ("sostenido", "almohadilla", o como quieras llamarle). Date cuenta de que el estilo se refiere a un sólo elemento de la página, no a varios. También podrías poner `#titulo` y se aplicaría al elemento que tenga `id="titulo"`, sea cual sea (en este caso es un `<div>`).

Otra forma de "seleccionar" elementos es intentar ser más específico. Por ejemplo, si todos los párrafos que hay dentro de un `<div>` son especiales, en vez escribir `<p class="especial">` para cada párrafo, define la regla CSS como `div p { .......... }`, que quiere decir lo mismo ("todos los `<p>` que estén dentro de un `<div>`"), y pon sólo `<p>` en cada párrafo.

Por cierto, ya has visto qué hace `class` e `id`. Aún hay otro atributo, `name`, que no hay que usar porque es lo mismo que `id`.

`<script>` y `<style>` requieren `type`, no `language`

Hay que poner `<script type="text/javascript">`. El atributo `language` no se usa (de todas formas cada uno lo escribía como quería, o no lo ponía porque iba igual). Lo

mismo con el `<style>`: si es CSS, pon `<style type="text/css">`.

Esta información sí que es útil, porque son tipos MIME estandarizados, no palabritas cualquiera.

Las cabeceras `<h1>`, `<h2>`, `<h3>`, ... son para cabeceras

No uses etiquetas de cabecera sólo porque el texto sale más grande. Sirven para definir la estructura del texto, y poder poner títulos de sección, subsecciones, etc. Los buscadores las usan para saber de qué se habla en una página, así que es importante usarlas sólo para lo que se han diseñado.

Cuando quieras cambiar el tamaño del texto, usa el CSS (propiedad `font-size`).

`<body>` no necesita atributos

Los atributos como `topmargin`, `leftmargin`, `marginheight`, `marginwidth`, `background`, `bgcolor`, `bgproperties`, `text`, `link`, `vlink`, `alink` son un ejemplo de lo mal que estaba Internet antes de CSS. Todos ellos se aplicaban al `<body>`, aunque algunos atributos fueron inventados por cierta empresa y no funcionaban igual en todos lados.

El caso es que todos ellos forman parte del diseño de la página (márgenes, fondos, colores, ...), y lo mejor es hacerlo con hojas de estilo. El `<body>` hay que acabar escribiéndolo como `<body>`, sin atributos (como la mayoría de etiquetas en HTML).

Conoce las etiquetas HTML

Hay muchas etiquetas para las cosas que salen normalmente en páginas web: direcciones, citas, código, referencias, variables, ... ¿Conocías `<abbr>`, `<acronym>`, `<address>`, `<bdo>`, `<cite>`, `<code>`, `<del>`, `<dfn>`, `<ins>`, `<kbd>`, `<q>`, `<samp>` o `<var>`?

Bastantes se comportan de la misma forma, pero al que lee el código le sirven para entender mejor la estructura del documento. También le van muy bien al diseñador, que podrá aplicar estilos diferentes a cada tipo de elemento sin tener que ir definiendo clases.

Puedes mirar las etiquetas aceptadas en la especificación de HTML 4.01.

No uses `<marquee>` ni `<blink>`

Son etiquetas propietarias (inventadas por empresas malas) y sirven para molestar. Una lo hace con texto que se mueve, y la otra hace que parpadee.

Por suerte, no van en muchos navegadores. Si quieres irritar a tus visitantes de una forma más efectiva, puedes usar imágenes (un GIF animado, por ejemplo), que eso sí que lo podrán ver... a menos que tengan un navegador que permita bloquear los banners con dos clics, como el mío.

Colores

Muy importante hacerlo bien. El navegador casi no los pone, así que tienes que ser tú

quien lo haga.

Si pones uno, ponlos todos

Ésta es la regla de oro que hay que seguir con los colores. Piensa en que no todos usan la misma configuración de colores que tú. Por ejemplo, a mí el negro sobre blanco me gusta sobre papel, pero en la pantalla del ordenador prefiero el blanco sobre negro, y es así como tengo configurado mi navegador.

¿Qué pasa entonces si tú pones un texto de color marrón? Que tú lo verías así, pero yo lo vería así, que cuesta de leer. La solución es sencilla: si vas a cambiar el color del texto en primer plano, cambia también el color del fondo. Así te aseguras de que quien pueda ver los colores, los vea todos bien combinados. En este caso, deberías asegurarte de que, donde sale el texto marrón, el fondo sea blanco.

Todos en el CSS

Todos los colores forman parte del diseño. No pongas algunos mediante HTML y otros mediante CSS, o te pasará el problema anterior.

Formatos para especificar colores

Hay varios. Por ejemplo, todas estas reglas son equivalentes (definen el mismo color):

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

La primera es con el nombre y se entiende bien, aunque tienes que saber qué nombres son los aceptados. La segunda está en formato RGB en hexadecimal, con un valor de 0 a f para cada color. La tercera permite afinar más y poner cada color no de 0 a f sino de 00 a ff (256 en vez de 16). La cuarta hace lo mismo en decimal, y la quinta con porcentajes.

Todo esto lo explica el apartado de colores de la especificación CSS.

En hexadecimal llevan el

No es opcional. Hay que poner `color: #0077ff;` y no `color: 0077ff;`.

Imágenes

No todos los navegadores las soportan, pero para los que sí, haz que las vean bonitas.

Cuándo usarlas

Yo divido las imágenes en dos tipos:

- Decorativas: complementan al texto y lo hacen más bonito.

- Necesarias: hacen falta para entender el texto. Por ejemplo, si contienen un mapa, una fórmula matemática, o algo que no puede explicarse con palabras.

Todo lo que digo aquí se aplica a cualquier imagen, pero en especial a las que son necesarias en una página. Éstas hay que tratarlas especialmente bien, porque son importantes y hay que dar alternativas a los que no puedan verlas.

Nunca uses imágenes para espaciar el contenido. Con CSS los márgenes se ponen en un momento y sin complicarse la vida.

Formatos gráficos

Hay pocas alternativas, y casi todas tienen algo malo. Lo que yo siempre recomiendo es:

- Para fotos o imágenes borrosas en las que salen trozos difuminados o no te importa perder calidad, usa el JPG. Ocupará poco, pero los detalles que pierdas en la conversión no los podrás recuperar.
- Para imágenes en las que hay zonas grandes del mismo color, sin trozos muy borrosos, como un cómic o una captura de pantalla, usa PNG. No se pierde calidad, y ocupa muy poco.
- Si la imagen tiene trozos transparentes o semi-transparentes, usa PNG. Soporta niveles de opacidad, que quedan muy bonitos.
- Si la imagen es animada: no hay muchas opciones buenas. Confío en que APNG será un buen formato cuando esté acabado, pero de momento habría que usar GIF animado. MNG es un buen formato para usar en Intranets y sitios donde podamos asegurarnos de que todos los navegadores tengan el soporte MNG activado.

Los demás formatos (PCX, GIF, BMP, PNM, TIFF, TGA, ...) son muy anticuados y poco útiles, además de ocupar demasiado para usarse en una página web.

Pon siempre el atributo `alt`

Todas las imágenes tienen que tener un texto alternativo, que se mostrará si el usuario tiene desactivadas las imágenes en su navegador (no quiere verlas) o si por alguna razón no puede (por ejemplo, si ha habido un error en la conexión o la imagen no existe, o también si el usuario es ciego). Además, mientras no hay ninguna imagen por mostrar porque se está cargando, el navegador pone este texto.

El texto alternativo hay que ponerlo con el atributo `alt`, por ejemplo `<img src="imagen3.jpg" alt="Foto de mi ordenador grande">`. Debe ser una descripción de la imagen (qué es la imagen), y no comentarios extra, ya que si se muestra el texto es porque la imagen no se puede ver.

Todas las imágenes importantes de la página deben llevar un buen atributo `alt`. Las imágenes decorativas o que no vale la pena explicarlas con palabras pueden llevar `alt=""` (vacío), pero el caso es que siempre hay que poner el `alt`.

Ese texto que sale al pasar el ratón se hace con `title`

Si quieres que al pasar el ratón por encima de una imagen salga un cuadrito con un texto, tienes que poner el atribute `title` (título de la imagen). No se hace con el atributo `alt` (ver apartado anterior) ya que ése sirve como sustituto de la imagen, para

cuando no se puede mostrar. El `title` es para añadir información sobre la imagen o comentarla. Naturalmente, puedes poner los dos.

Por ejemplo, `<img src="imagen3.jpg" alt="Foto de mi ordenador grande" title="Todos esos cables molestan mucho y cuestan de limpiar">.`

Usa el CSS para la altura, anchura y borde

En vez de usar los atributos `height`, `width` y `border`, haz algo como `<img src="imagen3.jpg"alt="" style="height:280px; width:210px; border: 1px solid black;">`, porque son cosas relacionadas con el diseño (y el CSS a lo mejor te permite usar clases para evitar repetir información).

Va bien que el navegador conozca la altura y anchura antes de cargar la imagen, porque así puede dibujar un cuadrito del tamaño exacto, en donde se colocará cuando esté cargada. Si no lo pones y la imagen es mucho más grande de lo que el navegador se había imaginado, todos los elementos de la página se reajustan para que quepa, y eso queda muy feo y molesta.

Enlaces

Supongo que es lo más importante en Internet. Si no fuera por ellos, no conoceríamos tantas páginas web.

Los buscadores los visitan (tenlo en cuenta)

Google y otros se pasean por las páginas web buscando enlaces, y cuando encuentran uno, añaden la dirección junto con las palabras con las que se ha hecho referencia a la página. Tú (el webmaster) puedes controlar qué información se mete en Google, y hacer que pasen cosas como ésta: la SGAE sale en primer puesto al buscar "ladrones". Y todo por crear enlaces como éste: ladrones.

Usa un texto descriptivo

Por la razón anterior, vigila el texto de tus enlaces. No hagas cosas como:

Aquí puedes ver mi artículo sobre deducción natural.

sino:

Puedes ver mi artículo sobre deducción natural.

porque no tiene ningún sentido querer que salga la web cuando alguien busca "aquí" en Google, sino que interesa que salga cuando busque "deducción natural".

No especifiques destino del enlace (`target`)

No tienes que decir cómo tiene que abrírsele al usuario cada enlace; él sabe mejor que tú cómo le gustan en cada momento (en pestañas, en ventana nueva, en popup, en la misma página, ...), y sabrá administrárselos. Si el visitante es tan novato que no sabe hacer eso, lo mejor que le puede pasar es que cada enlace se le abra en el mismo sitio

(todo lo demás es muy lioso), así que tú pon el enlace normal y ya está.

Los "enlaces" con JavaScript no son enlaces normales

Hay páginas en las que los enlaces no llevan a una dirección URI, sino que intentan ejecutar un código JavaScript. Eso hace que mucha gente no pueda entrar (como el robot de Google), y hace muy incómoda la navegación para los visitantes que sí que pueden verlos, pues impide eso de "Abrir enlace en nueva..." pestaña o ventana.

No cuesta nada poner un enlace normal, con su `href`. Hazlo así.

Tablas

No son necesarias, pero los que no conocen CSS las usan para cualquier cosa.

Casi no hacen falta

Las tablas (`<table>`) pocas veces son necesarias en Internet. Debes usarlas cuando tienes que mostrar unos datos en estructura tabular, o sea, organizados en filas y columnas. Por ejemplo... pues la famosa tabla periódica, o una tabla con correspondencias entre euros y pesetas. ¿Alguna vez has tenido que poner algo así en tu página? Yo, no muchas.

Nunca las uses para centrar, aplicar bordes, o separar objetos. Para todo eso está el CSS, que es mucho más cómodo, corto, y no lía a los navegadores.

Recuerda: únicamente para datos en estructura tabular.

Estructura de una tabla

En una tabla tienes que separar la cabecera de las filas normales, para que los navegadores lo sepan. Así, si alguien imprime (en papel) una tabla muy larga que ocupa varias páginas, el navegador sabrá repetir la cabecera al principio de cada página.

Si no conoces las etiquetas usadas para tablas (`<table>`, `<tbody>`, `<tr>`, `<th>`, `<td>`, `<caption>` y otras), mira la sección de tablas del HTML, o consulta una página más específica. Hay cosas interesantes, como el `<caption>`, que sirve para explicar de qué va la tabla (no todos pueden "echar un vistazo" para saber eso).

Ni `<table>` ni `<tr>` ni `<td>` necesitan atributos

Ni `width`, ni `height`, ni `bordercolor`, ni `background`, ni `bgcolor`, ... Todo eso forma parte del diseño y se hace con CSS. Las tablas y sus etiquetas correspondientes sirven para estructurar la información, no para adornarla.

Frames

Esto ya está anticuado, pero quería escribir un poco, porque una vez hablé con alguien que aún aseguraba que iban bien.

No hacen falta

Lo normal es que en una página no haga falta partir la pantalla en dos. Eso ya lo hará el usuario, si tiene un gestor de ventanas que le permita mover, redimensionar y organizar las ventanas. Aunque los usuarios no suelen hacer eso de ir moviendo las páginas de sitio por la pantalla... porque no hace falta.

Inconvenientes (*frames are evil*)

Muchos:

- No se pueden guardar en los marcadores ("Favoritos") fácilmente. No se guarda la página que estás viendo, sino la del `<frameset>`.
- El botón "Atrás" del navegador deja de comportarse como de costumbre. Lo mismo con el "Actualizar": si le das, te llevará a otra página distinta de la que estabas viendo.
- Es un caos intentar imprimir una página con frames.
- El tener varias zonas con barras de scroll puede volver locos a muchos usuarios. Peor aún si no hay barras de scroll y el contenido no cabe en la pantalla.
- Los buscadores se lían al visitar páginas con frames. Si no te lo crees, haz la prueba :-) ¿Por qué pasa? Porque rompes lo que siempre se ha seguido en Internet: cada página tiene *una* dirección, y cada dirección representa *una* página.
- Desperdician trozos de pantalla que algunos necesitan (como los que usamos resoluciones bajas).
- Dan problemas de seguridad: tú estás viendo una URI en la barra de direcciones, pero la página que estás visitando dentro del frame puede no tener nada que ver con esa dirección.

Hay muchas páginas que estudian todos estos aspectos con detalle. Intenta entender por completo cada punto, porque son problemas de usabilidad graves que debes evitar en tus webs.

"Ventajas" de los frames, desmentidas

Los novatos creen que:

- Hacen que la página sea más rápida, porque hay un trozo fijo que no tienen que cargar cada vez. Se olvidan de todo lo que ocupa el código de los frames, y del caché de los navegadores, que almacena las últimas imágenes visitadas y hace que no se descargue otra vez una imagen ya vista.
- Son fáciles de mantener, porque el mapa de la web y cada apartado están separados. No piensan en que tienen que añadir dos páginas nuevas sólo para los frames, y actualizarlas siempre a la vez.
- Dan usabilidad, ya que el menú está siempre visible en pantalla. Lo malo es que no se pueden asegurar de que esto pase siempre (con cualquier resolución), y quizás tampoco hayan pensado en que tener siempre visible el menú no es necesario.
- Permiten mantener la misma dirección en la barra de direcciones. Eso es malo (lee la sección anterior). A los visitantes les costará añadir una página a los marcadores.

No les veo más ventajas...

Qué usar entonces

No es necesario tener siempre el menú en cada página. Puedes tener una página de entrada con el menú, que te lleve a cada una de las secciones de la web.

Si necesitas repetir el mismo contenido en varias páginas, hay varios métodos, todos complicados o malos:

- Usa *Server-Side Includes* o un lenguaje como PHP, o CGI. El servidor web tiene que soportarlo.
- Incluye en cada página un `<script src="fichero.js" type="text/javascript">` en donde se haga un `document.write("código repetido");`. Esto equivale a no poner nada de ese código en todos los navegadores que no soportan JavaScript (que son muchos, sobre todo si cuentas a Google). Así que pon además enlaces de verdad.
- Copiar y pegar. Puede que sea lo más cómodo...
- Piensa un poco para hacer un programa (o método) que reemplace una frase mágica situada en cada `.html` por el código real que hay que repetir. Creo que ya hay programas que hacen eso.

Cuándo usar frames

Muy pocos casos justifican el tener que usar frames. Como dan tantísimos problemas de usabilidad y accesibilidad, ponerlos en Internet no tiene sentido.

Úsalos si quieres que los buscadores no indexen tu página, los usuarios no te puedan añadir a los marcadores, no tengan que imprimirse, vayan a verse en el mismo tipo de pantallas, y creas que no puede haber problemas de seguridad o de copyright. Este escenario podría ser una Intranet, aunque en ese caso valdría la pena instalar un lenguaje de servidor (ya lo he explicado más arriba).

JavaScript

Este tema es muy extenso. Puedes consultar la especificación de ECMAScript, que es como se llama el estándar.

Mejor usa scripts en el servidor

Si en tu servidor web puedes poner algún lenguaje como PHP, o CGIs (cualquier lenguaje), haz los programitas en el servidor. Es mucho mejor que trabaje el servidor que no los ordenadores de los clientes, porque no todos lo harán igual; y ni siquiera eso: puede pasar que algún usuario no quiera o no pueda ejecutar ningún tipo de script.

Si piensas en esto verás que la única forma de que todos los visitantes reciban el mismo trato es haciendo que prepare las páginas el servidor.

Los scripts han de hacer cosas cómodas para el usuario

Yo veo justificado usar JavaScript para ayudar al usuario en cosas que no se pueden hacer con HTML o CSS. Por ejemplo, que al cargar la página ya haya un cuadro de texto enfocado para empezar a escribir, o que si hay muchas casillas de selección juntas, haya una opción para "Seleccionar todas". Estas cosas ayudan, y el visitante las agradecerá. Para eso es para lo que hay que usar scripts.

Cuando ya no ayudan a la navegación, es cuando cambian la interfaz de la ventana, bloquean eventos (como el clic derecho), o estorban con cosas que se mueven. Todo esto hace la web incómoda, y es mejor evitarlo.

Que sea opcional

Los scripts no han de hacer cosas imprescindibles para ver bien la página. Son sólo para complementar y hacer la web más fácil de usar.

Así que si usas un menú JavaScript, asegúrate de que haya formas normales de llegar a cada sección (con los enlaces, `<a>`, como siempre).

No hagas *browser sniffing*

Con todo lo que he explicado, es fácil hacer que una web se vea bien en todos los navegadores, así que no hay necesidad de mirar mediante scripts qué navegador usa cada uno para darle un código u otro.

Flash

El Flash no es un estándar, aunque su especificación la pueden ver todos y por tanto cualquiera puede crear programas que usen el formato SWF.

Es una tecnología propietaria (lo contrario de estándar), porque su especificación la ha hecho únicamente Macromedia, y es suya (sólo ellos pueden cambiarla). Los estándares ISO o W3C se han hecho entre gente de todo el mundo, y aceptando comentarios y peticiones de los usuarios normales.

No es para sustituir al HTML o CSS

Varias veces he visto a expertos en CSS luchando contra expertos en Flash, discutiendo sobre qué es mejor y cuál hace más cosas. Mi opinión es que Flash permite hacer maravillas en una página web, y un montón de cosas que no se pueden hacer con CSS o HTML. Pero todo lo que hace CSS tampoco se puede hacer con Flash. Son cosas distintas: el CSS para hojas de estilo y el Flash para aplicaciones (sencillas o complicadas).

En lo que sí que les veo el parecido es en que sirven para complementar al contenido de la página. Pero no verás que nadie lo use para estructurar contenido ni para aplicar estilos a grupos de elementos de una página.

Ha de ser opcional ver una animación en Flash

En una página web lo más importante es que el contenido esté bien estructurado con HTML válido. Todo lo demás (CSS, imágenes, JavaScript, Flash, Java, ...) es para los privilegiados que lo puedan ver.

Por eso no puedes obligar a nadie a que para ver una página o acceder a otras necesite tener y usar el plugin de Flash (ni pedirle que tenga soporte CSS, ni JavaScript, etc.). Todo debe ser accesible desde el HTML, y el contenido en Flash es contenido extra destinado a quien pueda verlo.

Piensa que es muy normal no tener el plugin de Flash, aunque las estadísticas de Macromedia digan lo contrario. Hay muchos sistemas operativos para los que no existe visor de SWF, muchos navegadores que por razones técnicas no lo soportan, y muchos usuarios que por razones personales no quieren ver animaciones en Flash. Otros, como yo, lo tienen instalado pero desactivado por defecto, de forma que sólo se muestra si el visitante decide hacer clic en un botón que sale.

Cuándo usar Flash

Cuando haga falta. No lo uses para escribir textos largos; para eso está el HTML. Si lo haces, es parecido a no tener página web, porque es muy difícil llegar a algunos contenidos Flash. Suelen estar escondidos -tanto para humanos como para programas-, así que es mejor no poner información interesante ahí dentro.

Úsalo para animaciones vectoriales, o programitas interactivos sencillos. Casi siempre se ha usado para hacer animaciones que sean divertidas, y también para que las empresas de diseño hagan su propia web anunciando lo mucho que saben de Flash (y poco de accesibilidad).

Java

Otra tecnología propietaria (por ahora), pero muy potente y también algo lenta...

Cuándo poner applets Java en una web

Ten en cuenta que no todos tienen plugin de Java, así que no pongas cosas muy importantes.

Si sólo lo quieres para hacer una animación o un efecto especial bonito, es una mala elección, porque no todos lo podrán ver, y como es lento en cargar, suele molestar, sobre todo si no aporta nada útil. En vez de applets puedes usar imágenes bonitas, o GIF animados, o Flash.

El Java va bien para poder probar programas de ordenador en una web. Para programas muy complicados podrías decir "Bájate el `fichero.jar` y escribe `java -jar fichero.jar`", pero algunos ven más cómodo poder ejecutar el programa directamente desde la página.

Lo característico de casi cualquier programa es que es interactivo: necesita que el usuario haga algo, trabaja, y da una salida. Eso también se puede hacer en Flash o JavaScript, pero para cosas complicadas es mejor el Java por ser portable (multiplataforma, o al menos más que el Flash).

Así que, para programas interactivos, un applet de Java está bien. Si no es nada interactivo, ponlo en otro formato y evita el Java.