



La regla general dice que puede aprenderse cómo hacer una página web mirando el código creado por otros pero, lo normal es que si seleccionamos *Ver Código Fuente* en el navegador, nos encontraremos frente a frente con una madeja muy difícil de desatar y abandonaremos la idea diciéndonos: esto es imposible.

Este tutorial o como quiera llamarse tiene sólo un fin, aprender a hacer páginas web y a su vez, demostrar que es mucho más fácil de lo que parece.

Índice de contenido

A modo de introducción.....	4
Principios elementales.....	6
La estructura básica de una pagina web.....	8
Los párrafos.....	11
Textos especiales.....	15
Los vínculos.....	17
Las listas.....	20
Las imágenes.....	23
Las tablas.....	26
Creación de formularios.....	31
Controles para formularios.....	36
Insertar archivos.....	40
Las etiquetas audio y vídeo.....	42
Los scripts.....	44
Los estilos.....	46
Semántica y estructura.....	49
La etiqueta canvas.....	52
Las limitaciones de la tipografía.....	53
Referencias.....	56
¿Por qué Word no sirve para crear páginas web?.....	57
Los errores más comunes.....	59
La etiqueta META.....	61
Los atributos comunes.....	64
Etiquetas y atributos depreciados.....	66
Preguntas y respuestas sobre enlaces.....	68
Los formatos de las imágenes.....	72
Las miniaturas que no son miniaturas.....	74
Malditas tablas.....	75
Tabla de códigos Unicode.....	77
Códigos Unicode extendidos.....	80
Favicons: Ese dibujito que se ve en el navegador.....	82
Los estándares web.....	83
¿follow o nofollow?.....	85
Tabla de colores.....	86
Los formatos multimedia.....	89

A modo de introducción

La web no es más que un montón de documentos enlazados unos con otros que, para ser interpretados deben estar codificados de cierta manera, utilizando un lenguaje que todos los navegadores entiendan. Este lenguaje es llamado HTML (*HyperText Markup Language*).

El HTML es un lenguaje basado en etiquetas cada una de las cuales muestra, describe o agrega contenido al documento.

Para comenzar a hacer una página web, sólo son necesarios dos elementos:

1. Un procesador de texto cualquiera como el *block* de notas de Windows o cualquier otro que permita guardar los archivos como **Sólo Texto**, es decir, sin formatos de ningún tipo.
2. Un navegador.

El hecho que no sea necesario ningún programa especial es una de las características más importantes para el programador. Esto permite crear páginas web con cualquier PC y sistema operativo ya que: el código HTML sólo es texto.

Pero ¿no es mejor usar algún software que nos ayude?

Es cierto que existen programas que ayudan a escribir documentos HTML pero tengo una opinión bastante particular sobre este tema. Cada vez que me preguntan, siento que mi respuesta decepciona: no se necesita ningún programa especial, basta y sobra el *block* de notas de Windows o cualquier aplicación similar.

Pero, ¿no me conviene usar Dreamweaver, GoLive o Front Page? me han recomendado muchos programas gratuitos que pueden bajarse de internet; en serio, ¿tengo que usar el block de notas? ¿es una broma?

No, no lo es. El código HTML no es más que texto y por tanto, lo único necesario para escribirlo es un editor de texto sencillo; además, es conveniente comenzar a hacerlo de manera manual para comprender bien la estructura del lenguaje. Luego, una vez que hayamos pasado esta etapa básica, entonces, recién entonces, podemos utilizar herramientas como [Notepad++](#) o similares que nos facilitarán algunas tareas.

Hay una malsana costumbre, hacerle creer a la gente que los editores de tipo WYSIWYG (*What You See Is What You Get*, en inglés, "*lo que ves es lo que obtienes*") como Dreamweaver o GoLive son ideales para aprender a crear páginas web. Los venden con esta consigna pero, no es cierto.

No recomiendo utilizar los programas de tipo ya que algunas de las características avanzadas de estas herramientas pueden distraernos o confundirnos y generalmente, agregan códigos automáticos que terminan por ensuciar lo que hacemos, nos impiden analizarlo o nos dificulta encontrar problemas.

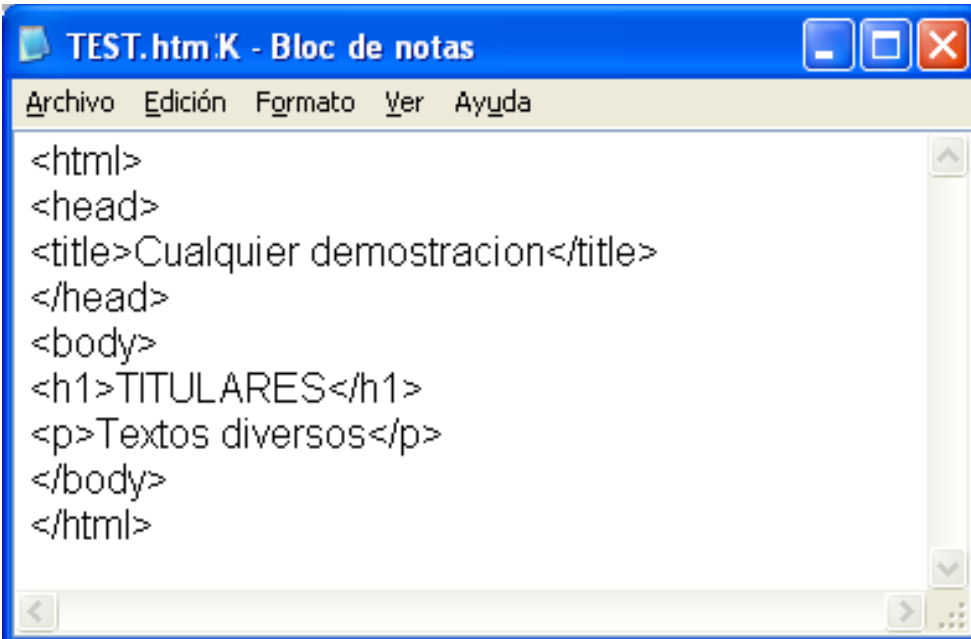
Algo parecido ocurre con una nueva generación de editores *online* o que pueden integrarse a páginas web como TinyMCE. El código resultante será siempre incierto y dependerá de cada uno de ellos. Pondremos un texto en negritas y uno escribirá ``, otro ``, otro ``

Jamás deben usarse procesadores de texto como Word u OpenOffice ya que lo único que se necesita es un archivo en texto plano y esos programas son PROCESADORES de textos lo que nos obligaría a convertir lo que escribimos para eliminar los formatos.

Si bien permiten crear páginas web sin escribir código HTML, no podemos aprender sin ver o entender que estamos haciendo y cuáles son nuestras alternativas. Como casi todos, hace años, intenté usarlos para hacer algo y fue frustrante no poder entender cómo poner un texto al lado de una imagen. Había decenas de opciones y menús pero, al no saber de qué hablaban, era imposible encontrar una respuesta.

Pero, en serio, ¿tengo que usar el block de notas?

Sí, por lo menos hasta que se haya entendido el concepto. Atrévanse. Las teclas `<` y `>` están ahí, muy cerca de la mano izquierda.



```
<html>
<head>
<title>Cualquier demostracion</title>
</head>
<body>
<h1>TITULARES</h1>
<p>Textos diversos</p>
</body>
</html>
```

Una vez que se escriben los códigos necesarios con un simple procesador de textos, el archivo se guarda con cualquier nombre y con extensión `.html` o `.htm` (en MS-DOS o cualquier sistema que sólo acepte tres letras en la extensión) y luego, se lo abre con el navegador.

VER REFERENCIAS [¿Por qué Word no sirve para crear páginas web?]

Principios elementales

El principio esencial del lenguaje HTML es el uso de las etiquetas (*tags*) y una etiqueta no es otra cosa que una palabra identificadora (en mayúscula o minúscula), delimitada por los caracteres < (mayor que) y > (menor que) con la siguiente estructura:

```
<etiqueta>
    [... contenido ...]
</etiqueta>
```

Por regla general, las etiquetas forman bloques que se inician con una etiqueta y finalizan con la misma etiqueta a la que se le antepone una barra inclinada (/)

Cada elemento se divide en tres partes: una etiqueta inicial, el contenido, y una etiqueta final. Todo lo que hay entre la etiqueta de apertura y la etiqueta de cierre se ve afectado por ella.

Otra característica básica es comprender que las etiquetas pueden anidarse, es decir, colocarse una dentro de la otra.

```
<etiqueta>
    <etiqueta>
        [... contenido ...]
    </etiqueta>
</etiqueta>
```

Y este concepto de etiquetas de apertura y cierre es la clave para entender el lenguaje. Como cualquier elemento puede contener a otros elementos, las etiquetas deben abrirse y cerrarse en el mismo orden en que fueron creadas así que la regla universal es: la última etiqueta abierta debe ser la primera en cerrarse.

Hay que imaginar que las etiquetas son cajas dentro de las cuales colocamos otras cajas que a su vez pueden tener otras cajas y así sucesivamente; esto, genera un orden que puede no ser muy evidente si no tenemos la precaución de escribirlo de modo prolijo. Para esto, es recomendable el uso de la tecla TAB de tal forma de poder visualizar cada bloque:

```
<etiqueta>
    <etiqueta>
        [... contenido ...]
    </etiqueta>
    <etiqueta>
        <etiqueta>
            [... contenido ...]
        </etiqueta>
    </etiqueta>
</etiqueta>
```

Hay ciertas etiquetas que son auto-suficientes y no requieren una etiqueta de cierre, en ese caso, en algunos sistemas se agrega la barra inclinada al final de esa única etiqueta:

```
<etiqueta/>
```

VER REFERENCIAS [Los errores más comunes]

Aunque solemos usar etiquetas simples, la mayoría de las veces, estas requieren datos adicionales llamados atributos y estos se colocan siempre en la etiqueta de apertura separados por un carácter espacio:

```
<etiqueta atributo1="valor" atributo2="valor">
    [... contenido ...]
</etiqueta>
```

Estos atributos son variados y dependen del tipo de etiqueta pero, por regla general constan de dos partes, su nombre y su valor que siempre va entre comillas.

Como todo, hay excepciones y nos encontraremos que ciertos atributos no requieren valor alguno ya que ese valor es un dato opcional; en ese caso, simplemente lo enumeramos.

```
<etiqueta atributo1="valor" atributo2>
    [... contenido ...]
</etiqueta>
```

Entonces, resumiendo:

1. la última etiqueta que creamos es la primera que debe cerrarse
2. indentar el código usando la tecla TAB para visualizar qué cosa esta dentro de que otra y de ese modo comprender cómo se verán afectados los contenido
3. nunca olvidarse que los atributos deben separarse con espacios y su valor debe estar entre comillas

La estructura básica de una pagina web

Una página web está compuesta por elementos (objetos) metidos unos dentro de otros (anidados). El elemento principal, el que contiene a todos los demás es el documento HTML.

Toda página web está comprendida entre dos etiquetas: `<html>` y `</html>`.

Esta etiqueta carece de atributos aunque desde la llegada del HTML5, se ha agregado uno llamado **manifest** cuyo valor debería ser la dirección de un documento cacheado de tal modo que la página pueda ser vista sin conexión a internet. Si alguien quiere ahondar el tema, la [Wikipedia](#) explica los detalles pero esto excede el uso clásico y no es algo cuyo uso se haya popularizado.

En ciertos casos específicos, cuando se trata de sitios que generan contenido de tipo XHTML como los blogs, se agrega el atributo `xmlns` que especifica la forma de interpretar el documento.

A su vez, el elemento `<html>` contiene otros dos, los elementos `<head>` y `<body>`. Esas son las principales zonas de cualquier página web, el encabezamiento, comprendido entre las etiquetas `<head>` y `</head>` y el cuerpo, comprendido entre las etiquetas `<body>` y `</body>`.

Dentro del encabezado `<head>` se coloca información genérica que afecta a toda la página pero que nada de esto es visible.

El elemento principal del `<head>` es el título, comprendido entre las etiquetas `<title>` y `</title>`. El título es un texto breve que describe el contenido y es el mismo que vemos cuando agregamos una página a los favoritos o marcadores:

```
<html>
  <head>
    <title>esta es mi página web</title>
    [... contenido ...]
  </head>
  <body>
    [... contenido ...]
  </body>
</html>
```

No son muchas las etiquetas que pueden ir dentro del `<head>` así que por ahora las enumeramos y seguimos adelante:

`<meta>` se utiliza para definir datos variados como el lenguaje o el autor y pueden tener uno o varios de estos atributos:

charset indica la codificación de caracteres utilizada
content es el valor que se debe asignar a otro atributo

http-equiv es el valor de header HTTP header de otro atributo
name es el nombre del elemento

VER REFERENCIAS [La etiqueta META]

<link> es la etiqueta que nos permite incluir documentos externos y tiene dos atributos obligatorios; uno llamado **rel** cuyo valor debe indicar el tipo de documento enlazado y otro llamado **href** cuyo valor es la URL del documento enlazado.

Posee más atributos pero todos son optativos y dependen de cada caso:

crossorigin indica la forma en que se trataran urls de dominios distintos

hreflang indica el lenguaje del documento enlazado

media indica el tipo de dispositivo para el cual se ha optimizado

sizes se usa cuando **rel="icon"** e indica el tamaño

type indica el tipo de documento

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

<base> indica la dirección URL base para todas las direcciones relativas que usemos y debe contener el atributo **href** que indica esa URL. Eventualmente, puede tener un atributo **target** para definir la forma en que se abrirá esa URL (**_blank**, **_parent**, **_self**, **_top**).

Por ejemplo:

<base href="http://misitio.com/images/" target="_blank">

Esas tres son EXCLUSIVAS del **<head>** y no pueden usarse en el **<body>** pero, hay otras dos que pueden agregarse indistintamente en ambos:

<script> es la etiqueta donde podremos agregar código en lenguaje JavaScript y **<style>** es la etiqueta donde podremos agregar las propiedades de estilo

Hay una etiqueta más que rompe todas las reglas y siempre se agrega al inicio.

Se trata de **<!DOCTYPE>** que, según la [W3C](#) defines el tipo de documento ya que toda página web debe identificar el tipo de código que se utiliza de tal forma que los navegadores comprueben su sintaxis.

Esta etiqueta es usual que sea agregada por los servicios de *blogs* y la mayoría de los programas de software que se utilizan para crear páginas web.

En la práctica, hoy en día, basta con definirla de esta forma: **<!DOCTYPE html>**

Entonces, nuestra primera página web podría tener este código básico donde notarán el uso de una etiqueta especial **<!-- y -->** que se utiliza para “comentar” un código es decir, indicarle al navegador que lo pase por alto. Todo lo que se encuentre entre **<!-- y -->** será ignorado.

```
<!DOCTYPE html>
<html>
  <head>
    <title>esta es mi página web</title>
    <meta charset="utf-8">
    <meta content="text/html; charset=UTF-8" http-equiv="Content-Type">
    <meta name="description" content="breve descripcion del contenido">
    <!-- [... otras etiquetas ...] -->
  </head>
  <body>
    <!-- [... contenido ...] -->
  </body>
</html>
```

La segunda parte elemental de una página web es el **<body>** y es allí donde se ubica el contenido visible (texto, imágenes, formularios).

La etiqueta **<body>** no suele tener atributos y cuando los tiene, siempre hacen referencia a la ejecución de algún tipo de función de JavaScript (**onafterprint**, **onbeforeprint**, **onbeforeunload**, **onerror**, **onhashchange**, **onload**, **onoffline**, **ononline**, **onpagehide**, **onpageshow**, **onpopstate**, **onresize**, **onstorage**, **onunload**).

Los párrafos

Ahora que conocemos la estructura básica de una página, podemos empezar a agregarle algo de contenido.

Una de las características principales de una página web es que, cuando ingresamos un texto, este se va a "acomodar" al tamaño de la ventana sin que tengamos que agregar ninguna instrucción o carácter especial.

Los navegadores no reconocen ni saltos de línea ni tabulaciones ni espacios en blanco extras a menos que se lo indiquemos de manera específica. Un código escrito así:

Esto lo escribo en una línea.
Y esto lo escribo en otra.
Y acá aquí dejo espacios en blanco.

Se verá así:

Esto lo escribo en una línea.Y esto lo escribo en otra.Y acá dejo espacios en blanco.

Si queremos separar los párrafos (o cualquier otro elemento), pero sin dejar una línea en blanco, debemos usar la etiqueta BR (*break*, o romper).

Esto lo escribo en una línea.

Y esto lo escribo en otra.

Y acá aquí dejo espacios en blanco.

Se verá así:

Esto lo escribo en una línea.
Y esto lo escribo en otra.
Y acá dejo espacios en blanco.

Pero, aún no vemos los espacios extras porque para eso debemos indicarlo de forma explícita utilizando un código especial: ` `; (*non-breaking space*).

Y acá de&nbps&nbps&nbpstres espacios en blanco.

Si bien podríamos agregar ese contenido tal cual se muestra en el ejemplo, una regla elemental es que **TODO** debe estar dentro de una etiqueta que indique qué es.

Para escribir párrafos, disponemos de la etiqueta `<p></p>` que, de modo automático, creará esos saltos de línea ya que se trata de lo que llamamos una etiqueta de bloque:

<p>este es un párrafo</p>

<p>y este es otro, entre ambos se verá un salto de línea</p>

¿Y qué es eso de la etiquetas de bloque?

Una de las claves para entender cómo funciona una página web es darse cuenta que todas las etiquetas son rectángulos y todo su contenido se encuentra dentro de una de ellas, lo sepamos o no lo sepamos, lo hayamos indicado o no.

Como todas, una etiqueta de bloque también es un rectángulo pero tiene una característica; sin importar su contenido, por defecto, ocupa todo el ancho de la ventana del navegador y es dinámica, si esa ventana se reduce o amplía, el contenido se adapta al ancho.

Al ocupar siempre todo el ancho disponible, cualquier otra etiqueta se mostrará debajo de esta, con un salto de línea.

La etiqueta `<p>` ya no posee atributos especiales, sólo admite los atributos comunes a todas pero esto, no siempre fue así por lo que es fácil confundirse.

VER REFERENCIAS [Los atributos comunes]

Es muy probable ver que aún se usa el atributo `align` para indicar la forma en que se alineará ese párrafo (a la izquierda, a la derecha o centrado) pero este atributo ya no es soportado por el HTML5 y en su lugar, se recomienda definir reglas de estilo (CSS).

Depreciado es una palabra fea y no se la deseo como calificativo a nadie. Ser despreciable es malo pero ser depreciado es peor. La [W3C](#), la organización que establece las reglas tiene una lista de etiquetas y atributos "depreciados", es decir, cosas que nos recomiendan no utilizar y nos aconsejan que coloquemos otros códigos alternativos. Muchas de esas etiquetas y atributos ya no se usan hace tiempo pero otras siguen dando vueltas por ahí.

VER REFERENCIAS [Etiquetas y atributos depreciados]

EL texto de los párrafos puede ser formateado de dos maneras, utilizando reglas de estilo CSS o con algunas etiquetas especiales que son fácilmente reconocibles ya que son similares a las que podemos aplicar en un procesador de textos.

La etiqueta `` muestra el texto en negritas (*bold*) pero tiene una etiqueta similar pero que se recomienda utilizar cuando lo que se pretende es que ese texto se enfatice:

`este texto es importantes`

La etiqueta `<i></i>` muestra el texto en cursiva (*italic*) y también tiene una etiqueta similar que se recomienda usar:

`este texto se enfatiza`

La etiqueta `<s></s>` muestra el texto tachado.

La etiqueta `<u></u>` muestra el texto subrayado.

Ninguna de ellas tiene atributos especiales y todas pueden aplicarse individualmente o combinarse:

`<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod <u>incidunt</u> ut laoreet dolore magna aliquam erat volutpat.</p>`

En matemáticas se utilizan índices y subíndices, estos se consiguen mostrar con las etiquetas `<sup>` y `<sub>` respectivamente. Por ejemplo, si quisiéramos escribir m² para indicar metros cuadrados:

`<p>m²</p>`

y para escribir V_x:

`<p>V_x</p>`

Para controlar los cambios en el tamaño de las fuentes de texto existían dos etiquetas de uso frecuente que permitían aumentarlo o disminuirlo:

`<p>texto normal <big>ABCDEFGH</big> y más grande</p>`
`<p>texto normal <small>ABCDEFGH</small> y más pequeño</p>`

Curiosamente a `<big>` se la considera una etiqueta depreciada pero `<small>` no.

Para terminar, hay algunas etiquetas especiales que no se utilizan para dar formato sino para identificar el tipo de texto.

abbr se usa para indicar que el contenido es una abreviatura

address indica información de contacto del autor

cite se utiliza cuando el texto es una cita o una referencia

code el texto es un fragmento de código

del indica que un texto se ha eliminado o ya no debe tenerse en cuenta

dfn el texto es la definición de un término

ins se usa para indicar que un texto se ha insertado en el documento

kbd indicamos que el texto debería ser introducido por el usuario

mark resalta parte de un texto

q se usa para indicar que el texto es una cita corta

ruby texto extra para indicar pronunciación (se usa junto con `<rp>` y `<rt>`)

time se usa para indicar una fecha u hora

samp se usa para indicar estamos mostrando la salida de un programa, script, etc

var el texto es una variable o un argumento

wbr indica que en el texto debería haber un salto de línea

Las etiquetas `<bdi>` y `<bdo>` están relacionadas con la dirección del texto (de izquierda a derecha o de derecha a izquierda). Su uso es muy limitado.

La mayoría de estas etiquetas se muestran de manera diferente según el navegador. Esto se debe a que, en general, al no ser de uso común, no se les asigna un formato y utilizan el predeterminado.

Las etiquetas ``, `<ins>` y `<q>` pueden incluir el atributo `cite` cuyo valor sería la dirección URL de donde se tomó el texto citado.

Las etiquetas `` e `<ins>` pueden incluir el atributo `datetime` indicando la fecha.

Algunos ejemplos:

```
<abbr title="Federal Bureau of Investigation">FBI</abbr>
```

```
<acronym title="North Atlantic Treaty Organization">NATO</acronym>
```

```
<p>el periódico <cite>Noticias</cite> recomendó la película Titanic.</p>
```

```
<p>Albert Einstein dijo, <q>Dios no juega a los dados.</q></p>
```

```
<code>
function saludar() {
    alert("HOLA");
}
</code>
```

```
<del cite="URL_pagina.html" datetime="01-01-2016">
  <p>Este es el párrafo actualizado.</p>
</del>
```

```
<ins cite="URL_pagina.html" datetime="01-01-2016">
  <p>Este es el párrafo insertado.</p>
</ins>
```

VER REFERENCIAS [Los estándares web]

Textos especiales

Los llamados encabezados también son textos pero tienen características especiales. El formato por defecto de los navegadores los muestra con un tamaño diferenciado del resto pero, esta no es su razón de ser ya que, en realidad, debemos utilizarlos para indicar la importancia o jerarquía de algo.

Los posibles encabezados son seis: <h1>, <h2>, <h3>, <h4>, <h5> y <h6>

Cada una de estas etiquetas mostrará el texto comprendido entre ellas con un tamaño distinto donde **<h1>** será el mayor y **<h6>** el menor.

En principio, una página web sólo debería tener una etiqueta `<h1>` y esa etiqueta debería ser el título principal, el nombre del sitio o del artículo sin embargo, esta regla clásica está siendo cuestionada.

De todos modos, si hay subtítulos usaríamos **<h2>** y así sucesivamente pero, sin abusar de ellos; si deseamos que el texto que queremos agregar se vea “grande” pero es un elemento intrascendentes, deberíamos usar una etiqueta **<p>** y definir una regla de estilo a gusto.

La etiqueta `<hr>` no es un texto, simplemente es una línea horizontal y tampoco debería usarse como elemento estético; sólo debería emplearse para separar contenidos temáticos.

Posee un atributo opcional con el cual es posible controlar su ancho ya que por defecto ocupará todo el ancho de la ventana; por ejemplo:

Ya habíamos dicho que el HTML tiene una particularidad, los espacios en blanco adicionales se eliminan sistemáticamente es decir, si escribimos:

`<p>Hola` `Adiós</p>`

Lo que veremos será:

Hola Adiós

Toda esa separación adicional será ignorada y si queremos separar una palabra de la otra, tenemos que recurrir a caracteres especiales como ` ` (*on-breaking space* o *no-break space*) que es el llamado *hard space* o *fixed space* (espacio fijo).

[illegible]

También podemos usar otra variante, escribiendo ** ** y si queremos reproducirlo con el teclado: Ctrl+Mayús+Espacio o Ctrl+Espacio.

Sea como sea, el resultado es un código confuso y da demasiado trabajo. Entonces, podemos recurrir a la etiqueta `<pre></pre>` (*PREformatted*) que nos permite formatear cualquier texto que se encuentre dentro de ella.

Por defecto, el texto que se encuentre dentro de ella aparecerá con una fuente de espaciado fijo (tipo *Courier*) pero lo más importante es que se respetarán los espacios en blanco y saltos de línea, tal y como los hayamos escrito en nuestro documento. Es decir, si escribimos:

```
<pre>
este texto
  preformateado
    se mostrará
      tal y como lo escribamos
</pre>
```

Se verá así:

```
este texto
  preformateado con PRE
    se mostrará
      tal y como lo escribamos
```

No sólo podemos usar espacio, también podemos usar TABs horizontales (`	`) para crear tablas sencillas:

La etiqueta `<blockquote></blockquote>` la utilizamos para destacar una cita textual dentro del texto general y posee un atributo opcional llamado `cite` donde es posible indicar la dirección URL de donde hemos tomado la cita.

Su sintaxis es sencilla y como las anteriores, los navegadores las muestran con un formato por defecto (fuente, márgenes, etc) y, por lo general, tratamos de diferenciarla gráficamente del resto de la página definiendo alguna regla de estilo personal.

“

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi.

”

Los vínculos

La característica más destacada de internet es la posibilidad de unir distintos documentos por medio de hipervínculos que es eso que comúnmente llamamos vínculos.

Los enlaces (hipertexto) son zonas (generalmente palabras, frases, imágenes o botones) que nos permiten navegar, ir de un lado a otro, con un simple *click* del ratón.

Para crear vínculos se usa la etiqueta `<a>` (*Anchor*) y en general tienen la siguiente estructura:

```
<a href="direccion_url">un texto cualquiera</a>
```

Donde podemos ver un atributo `href` que en la práctica es obligatorio ya que es donde indicamos que debe ocurrir cuando se hace *click* en él.

La dirección URL es el destino del enlace y por defecto, el texto visible tiene un color que lo diferencia y se muestra subrayado.

Podemos distinguir varios tipos de enlaces:

1. enlaces dentro de la misma página
2. enlaces a otra página dentro de nuestro sitio
3. enlaces a una página externa
4. enlaces a alguna función JavaScript

Los más sencillos de comprender son los enlaces a una página externa (una página que está en un servidor distinto al nuestro) y por lo tanto, necesitamos conocer su dirección URL completa (*Uniform Resource Locator*).

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

Como se ve hemos usado la dirección completa pero, en los últimos tiempos, se recomienda omitir el protocolo (`http:`, `https:`) y si optamos por esto, hay que tener en cuenta que la URL debe comenzar con `//` :

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

Lo más frecuente es que tengamos varias páginas que deberían enlazarse entre si y los enlaces a otra página dentro de nuestro sitio son similares a lo anterior.

```
<a href="pagina.html">Ir a la Página 2</a>
```

En este caso estamos suponiendo que ambas páginas están en el mismo directorio pero, generalmente, los sitios web se organizan de manera similar a como lo hacemos con nuestra PC: un directorio principal, y subdirectorios auxiliares.

Si la página a la que se quiere saltar está en un subdirectorio, la referencia debería ser:

```
<a href="subdirectorio/pagina2.html">Ir a la Página 2</a>
```

Y a la inversa, si la página con el enlace está en un subdirectorio y queremos saltar a un directorio anterior:

```
<a href=" ../pagina2.html">Ir a la Página 2</a>
```

A veces, en el caso de páginas extensas, es interesante dar un “salto” desde una parte a otra dentro de la misma y para conseguir esto, debemos identificar el destino, “marcándolo” con una etiqueta. Esto es lo que se denomina anclaje.

Un anclaje es un vínculo especial que nos permite desplazarnos a un lugar específico dentro de la misma página.

El código de este enlace puede estar en cualquier parte y, el destino, lo que llamamos ancla, es una marca en un punto determinado.

El anclaje lo creamos con la etiqueta `<a>`, como cualquier otro vínculo:

```
<a name="mimarca"></a>
```

A veces, se coloca un texto pero no se mostrará como vínculo, el navegador lo mostrará como si fuera un texto cualquiera:

```
<a name="mimarca">ejemplo</a>
```

Como se ve, la diferencia básica es que no se utiliza el atributo `href` sino el atributo `name` que contiene el nombre único con el que identificaremos la marca (el ancla).

El vínculo se escribe como uno normal donde el atributo `href` contiene el símbolo numeral y luego el nombre del anclaje:

```
<a href="#mimarca">ir al ejemplo</a>
```

Al pulsar en este vínculo, se salta al lugar de la página donde se encuentra el anclaje.

También puede crearse un vínculo a un ancla de otra página. Para eso, se procede de la misma manera pero agregando la dirección URL:

```
<a href="URL_pagina#mimarca">ir al ejemplo</a>
```

Este tipo de vínculo tiene una variante, utilizar los atributos `href` y `name` en la misma etiqueta:

```
<a href="URL_pagina" name="nombre">ir al ejemplo</a>
```

Por último, hay ocasiones en las que deberemos utilizar los vínculos para ejecutar alguna instrucción y no para dirigirnos a una página. En estos casos, lo que utilizamos son los llamados eventos de JavaScript para llamar a una función. Por ejemplo:

```
<a href="#" onclick="alert('HOLA');">haga click acá</a>
```

Por lo general, los ejemplos que encontramos en la red nos dicen que el atributo **href** debe contener el carácter **#** (*pound*), pero esto no es siempre así; en determinadas condiciones no funciona; la función se ejecuta pero además, la página parece recargarse o hacer un *scroll* a su inicio.

La alternativa es colocar como atributo, el valor: **javascript:void(0);**

```
<a href="javascript:void(0);" onclick="alert('HOLA');">haga click acá</a>
```

Esta instrucción, le indica al navegador que el vínculo “apunta a ninguna parte” y, por lo tanto, sólo debe ejecutarse la función y no moverse de ahí.

Una solución alternativa es agregar al evento **return false** que informa al navegador que la acción de evaluar el atributo **href** no debe ser ejecutada y, que no hay nada más que hacer.

```
<a href="#" onclick="alert('HOLA');return false;">haga click acá</a>
```

Ademas del atributo **href**, los vínculos pueden tener otros que son optativos:

download se utiliza para indicar que el enlace es un archivo que se descargará y si establecemos un valor, este será el nombre del archivo descargado:

```
<a href="misitio.com/una_imagen_de_ejemplo.jpg" download="mi-imagen">
```

hreflang indica el lenguaje del documento enlazado

media indica el tipo de dispositivo para el cual fue optimizado el documento enlazado

rel indica la relación entre la página actual y la página enlazada (alternate, author, bookmark, help, license, next, nofollow, noreferrer, prefetch, prev, search, tag).

target indica la forma en que se abrirá (**_blank**, **_parent**, **_self**, **_top**)

VER REFERENCIAS [Preguntas y respuestas sobre enlaces]

Las listas

Las listas nos permiten agregar información de manera más clara. Hay tres tipos diferentes: las listas no-ordenadas (no numeradas), las listas ordenadas (numeradas) y las listas de definiciones.

Las listas no-ordenadas (*Unordered Lists*) sirven para mostrar cosas que no necesitan ir precedidas por un número. La etiqueta a utilizar es `` y, cada ítem en la lista se agrega con la etiqueta ``

```
<ul>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
  <li>item 4</li>
</ul>
```

Las listas ordenadas (*Ordered Lists*) sirven para mostrar datos en un orden determinado. Los ítems se agregan igual que en el caso anterior, la diferencia es que utilizamos la etiqueta `` para enmarcar todo. El resultado será el mismo pero cada ítem estará precedido por un número correlativo.

```
<ol>
  <li>item 1</li>
  <li>item 2</li>
  <li>item 3</li>
  <li>item 4</li>
</ol>
```

Las listas de definición se utilizan para glosarios. Toda la lista debe ir entre las etiquetas `<dl></dl>` (*Definition List*). A diferencia de las anteriores, cada renglón tiene dos partes: el nombre de la cosa a definir (*Definition Term*) se coloca en la etiqueta `<dt></dt>` y la definición en si misma (*Definition Definition*) en la etiqueta `<dd></dd>`

```
<dl>
  <dt>palabra 1</dt>
  <dd>la definición de la palabra 2</dd>
  <dt>palabra 2</dt>
  <dd>la definición de la palabra 2</dd>
</dl>
```

La etiqueta `` tiene dos atributos opcionales:

reversed indica que el orden de la lista sera descendiente

start indica el valor inicial de la lista (por defecto es 1)

La etiqueta `` puede tener el atributo **value** indicando el valor del elemento.

Al igual que casi todas las etiquetas HTML, las listas pueden anidarse, es decir, ponerse una dentro de otra:

```
<ul>
  <li>animales
    <ul>
      <li>mamíferos
        <ul>
          <li>vaca</li>
          <li>león</li>
        </ul>
      </li>
    </ul>
    <ul>
      <li>reptiles
        <ul>
          <li>sapo</li>
          <li>serpiente</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>vegetales
    <ul>
      <li>cebolla</li>
      <li>pino</li>
    </ul>
  </li>
</ul>
```

Las etiquetas ``, `` y `<dl>` forman BLOQUES, visualmente, crean un salto de línea tal y como lo hace la etiqueta `<p>`

El atributo **type** define el gráfico que se muestra y por defecto es un *bullet* pero los navegadores los muestran de formas diferentes o simplemente ignoran ciertos tipos por ese motivo es que ha sido depreciado y se recomienda utilizar CSS.

Hay una etiqueta extra llamada `<menu></menu>` que resurgió de sus cenizas ya que había sido depreciada pero, la versión del HTML5 la ha rescatado y re-definido.

Se utiliza para crear la lista de un menú y los items utilizan la etiqueta `<menuitem>`. Un ejemplo:

```
<menu type="context" id="mimenu">
  <menuitem label="titulo_1" onclick="funcion1();"></menuitem>
  <menu label="subtitulo">
    <menuitem label="subtitulo_A" onclick="algo_A();"></menuitem>
    <menuitem label="subtitulo_B" onclick="algo_B();"></menuitem>
  </menu>
  <menuitem label="titulo_2" onclick="funcion2();"></menuitem>
</menu>
```

Estos son los atributos que pueden contener:

label contiene el texto que se visualizará.

type indica el tipo de menú (*list*, *toolbar*, *context*).

onshow es una función JavaScript optativa que se ejecutará cuando el elemento se muestre como menú contextual.

Las listas son uno de las etiquetas más comunes que usamos en una página web. Cuando comenzamos a aprender este lenguaje, suele parecernos que son algo “poco interesante” y a decir verdad, durante mucho tiempo fue así porque ¿para qué puede servir una lista si no es para hacer listas?

En los últimos tiempos, vaya uno a saber por qué, las listas comenzaron a tener un papel importante; alguien las sacó del olvido y se han transformado en las nuevas estrellas de las web ya que se usan para muchas cosas.

Sin embargo, hay que evitar el abuso ya que puede causarnos problemas porque son etiquetas que tienen una serie de propiedades por defecto y porque los navegadores suelen interpretarlas de manera diferente.

Las imágenes

La forma de incluir imágenes en nuestras páginas es muy similar a como lo hacemos con los enlaces a otras páginas. La única diferencia es que, en lugar de indicar el nombre y la localización de un documento HTML, se le indica el nombre y la localización de un archivo que contiene una imagen.

La etiqueta que utilizamos es `` y el atributo obligatorio es `src` (*image source*, fuente de la imagen):

```

```

El formato de la imagen es, en términos generales, irrelevante aunque normalmente se utilizan los formatos GIF, JPG y PNG.

VER REFERENCIAS [Los formatos de las imágenes]

Otro atributo importante es `alt` con el cual podemos introducir una palabra o una frase breve que describa la imagen.

Si bien se puede omitir, es la manera por la cual un navegador de sólo texto puede acceder a ellas o, por lo menos, hacerse una idea de lo que pretendemos mostrar. Además, como es posible que por alguna razón una imagen no pueda cargarse y muchas veces se las utiliza como enlace a otras páginas, es fundamental que mostremos algún texto alternativo que indique su destino

```

```

Las reglas respecto a la forma de ingresar el destino son las mismas que las que rigen para los enlaces. Si no se indica nada quiere decir que el archivo está en el mismo directorio que el documento HTML. Si la imagen está fuera de nuestro sitio, se debe indicar la URL o dirección completa.

Un aspecto muy importante a tener en cuenta cuando utilizamos imágenes es su tamaño. Cuanto más grande sea el archivo, más tiempo de carga será necesario y dado que los navegadores leen y ejecutan de manera secuencial, al encontrarse una etiqueta ``, se inicia la carga de la imagen y recién cuando termina se continúa con el resto de la página. Esto es así por una simple razón, el navegador desconoce el tamaño de la imagen y, por lo tanto debe cargarla por completo para saber dónde continuar.

Una forma de minimizar este efecto es utilizar otros dos parámetros, `width` (ancho) y `height` (alto) para indicarle al navegador cuál es el tamaño de la imagen y de esta forma pueda "reservar" el espacio necesario y continuar con la carga del resto de la página al mismo tiempo que va cargando la imagen.

```

```

Esto conviene hacerlo con todas, incluso con las más pequeñas (iconos, botones, etc), para que no haya ninguna interrupción en el proceso de carga del documento.

Pero debemos prestar atención a algo fundamental, usar **width** y **height** sólo muestran la imagen de cierto tamaño pero no la re-dimensionan.

VER REFERENCIAS [Las miniaturas que no son miniaturas]

En todos los casos, es esencial minimizar la demora en la carga de las imágenes y para eso, debemos hacer un uso razonable de las herramientas que nos proveen los programas gráficos:

1. reduciendo su tamaño (menos pixeles = menos kilobytes para cargar)
2. reduciendo el número de colores
3. simplificándolas o comprimiéndolas

Cuando se carga la imagen de una página, esta queda almacenada en el caché. Por lo tanto, si esta misma imagen se utiliza en otras páginas no será requerida al servidor para ser cargada de nuevo. Por ello, siempre que se pueda, es conveniente repetir la misma imagen para los botones, los íconos, las barras de separación, etc.

Otra posibilidad que nos dan las imágenes es la de utilizarlas como enlace a otra página. La forma de hacerlo es crear un enlace normal y reemplazar el texto explicativo por la imagen:

```
<a href="mipagina.com"></a>
```

Por defecto, si la imagen es un vínculo, estará rodeada de un rectángulo del mismo color que los enlaces. Si no se desea que aparezca, hay que indicarlo utilizando CSS o el atributo (depreciado) **border**:

```
<a href="mipagina.com"></a>
```

Hay tres atributos optativos que pueden usarse combinados con JavaScript:

onabort ejecuta una función en caso de interrumpirse la carga

onerror ejecuta una función en caso de producirse un error en la carga

onload ejecuta una función cuando la imagen ha sido totalmente cargada

Aunque están desuso, un tipo especial de imagen son los llamados mapas. Un mapa es una imagen única desde la cual podemos para enlazar a varias páginas, yendo a una u otra según el área en donde se haga *click* y esto se consigue utilizando las etiquetas **<map>** y **<area>**

Para crear un mapa, debemos partir de una imagen, y utilizar código HTML para crear las áreas (zonas activas o *hotspots*).

Para definir un área activa rectangular, necesitamos conocer las coordenadas de su ángulo superior izquierdo y las de su ángulo inferior derecho. Estas coordenadas las obtenemos con cualquier programa gráfico.

Supongamos que la primera zona activa tiene estas dos coordenadas (a1,b1) y (c1,d1) y que la segunda zona activa tiene las coordenadas (a2,b2) y (c2,d2), el código resultante sería este:

```
<map name="mimapa">
  <area shape="rect" coords="a1,b1,c1,d1" href="URL_destino1">
  <area shape="rect" coords="a2,b2,c2,d2" href="URL_destino2">
  <area shape="DEFAULT">
</map>

```

Cuando hagamos *click* en la parte superior de la imagen, nos redirigiremos a la página 1, y haciendo *click* más abajo, a la pagina 2.

El atributo **name** de la etiqueta **<map>** es cualquiera que se nos ocurra y sirve para identificar el mapa.

La etiqueta **<area>** posee tres atributos básicos:

coords son las coordenadas del área.

href es la dirección URL del vínculo que queremos abrir

shape especifica la forma de cada área (rec, circle, polygon)

<area shape="DEFAULT"> es una etiqueta que define el área completa del mapa, indicando que su forma es la que tiene por defecto (rectangular). Si se quisiera que el mapa no abarcara la totalidad de la imagen, o incluso que tuviera una forma distinta, habría que indicarlo aquí, en lugar de la instrucción **DEFAULT**.

La etiqueta **** final es similar a las usadas anteriormente excepto que se le agrega un atributo nuevo, **usemap** que indica que es el mapa definido anteriormente con ese nombre.

Las tablas

Las tablas en HTML son una de las formas más comunes de tabular el contenido alineándolo en filas y columnas.

Sin embargo, algunos afirman que *“las tablas no son necesarias, pero los que no conocen CSS las usan para cualquier cosa ... pocas veces son necesarias en Internet. Debes usarlas cuando tienes que mostrar unos datos en estructura tabular, o sea, organizados en filas y columnas. Nunca las uses para centrar, aplicar bordes, o separar objetos. Para todo eso está el CSS, que es mucho más cómodo, corto, y no lío a los navegadores.”*

Son muchos los desarrolladores que han contraído una especie de fobia por este tipo de etiquetas pero me voy a atrever a disentir un poco. Las tablas, como cualquier otro elemento del lenguaje no son más que una herramienta y serán tan “buenas” o tan “malas” como quién las opere. En mis manos, un martillo termina transformando cualquier cosa en un dedo sangrante, en manos de Miguel Ángel, convertirá un pedazo de piedra en el David. La diferencia está entre él y yo, no es culpa del martillo.

Mi humilde consejo es: usen la herramienta que les resulte más cómoda y mantengan los ojos abiertos para ver qué puede mejorarse.

No cabe la menor duda que las tablas son cómodas, fáciles de usar y sobre todo, fáciles de entender ya que podemos asociarlo mentalmente con una hoja cuadriculada y “diseñar” cosas relativamente complejas sin necesitar grandes conocimientos. Si a esto le unimos las bondades del CSS, no serán tablas semánticamente correctas pero los resultados serán más que aceptables.

VER REFERENCIAS [Malditas tablas]

En fin, olvidemos la filosofía. Una tabla es un conjunto de filas y columnas; la intersección de una fila y una columna determinan una celda, un lugar donde puede colocarse un dato y la etiqueta general que crea una tabla es `<table>`, las filas se crean con la etiqueta `<tr>` (*Table Row*) y las columnas con la etiqueta `<td>` (*Table Data*) así que esto, generaría una tabla de dos filas con tres columnas:

```
<table>
  <tr>
    <td>fila 1 columna 1</td>
    <td>fila 1 columna 2</td>
    <td>fila 1 columna 3</td>
  </tr>
  <tr>
    <td>fila 2 columna 1</td>
    <td>fila 2 columna 2</td>
    <td>fila 2 columna 3</td>
  </tr>
</table>
```

También podemos definir que la primera fila sirva de encabezado; para esto es que existe la etiqueta `<th>` (*Table Header*)

```
<table>
  <tr>
    <th>encabezado columna 1</th>
    <th>encabezado columna 2</th>
    <th>encabezado columna 3</th>
  </tr>
  <tr>
    <td>fila 1 columna 1</td>
    <td>fila 1 columna 2</td>
    <td>fila 1 columna 3</td>
  </tr>
  <tr>
    <td>fila 2 columna 1</td>
    <td>fila 2 columna 2</td>
    <td>fila 2 columna 3</td>
  </tr>
</table>
```

Una etiqueta alternativa, permite añadir un título general a toda la tabla, un texto encima de esta que indica cuál es su contenido. Se consigue con la etiqueta `<caption>`

```
<table>
  <caption>titulo de la tabla</caption>
  ... el resto de las etiquetas ...
</table>
```

Una serie de etiquetas opcionales se utilizan para agrupar todo y diferenciar los distintos sectores de la tabla. `<col>` y `<colgroup>` se usan para agrupar columnas, mientras que `<thead>` (cabecera de tabla), `<tfoot>` (pie de tabla) y `<tbody>` (cuerpo de tabla) se usan para agrupar las filas.

```
<table>
  <caption>titulo de la tabla</caption>
  <thead>
    <tr><th>encabezado columna</th></tr>
  </thead>
  <tfoot>
    <tr><th>pie columna</th><th>
  </tfoot>
  <tbody>
    <colgroup>
      <col>
    </colgroup>
    <tr>
      <td>contenido</td>
    </tr>
  </tbody>
</table>
```

¿Qué pasa si colocamos un número distinto de celdas en cada fila? El navegador dibuja la tabla y la completa con espacios en blanco.

```
<table>
  <tr>
    <th>columna 1</th>
    <th>columna 2</th>
    <th>columna 3</th>
  </tr>
  <tr>
    <td>fila1-celda1</td>
    <td>fila1-celda3</td>
  </tr>
  <tr>
    <td>fila2-celda1</td>
    <td>fila2-celda2</td>
    <td>fila2-celda3</td>
  </tr>
</table>
```

Ese espacio vacío no quedará en el medio (que sería la celda faltante) sino al final de la fila.

Hubo un tiempo en que las etiquetas de tablas poseían una gran cantidad de atributos opcionales pero el HTML5 ha hecho que eso quede en el olvido. Ya no se recomienda usar **abbr**, **align**, **axis**, **background**, **bgcolor**, **border**, **cellpadding**, **cellspacing**, **char**, **charoff**, **color**, **frame**, **rules**, **scope**, **nowrap**, **valign** ni **width**.

En las etiquetas **<td>** y **<th>** puede usarse el atributo *headers* para indicar la celda de encabezado relacionada. También hay un par de atributos que permiten que una celda se “expanda” abarcando otras.

colspan indica la cantidad de columnas a expandir

rowspan indica la cantidad de filas a expandir

Este es un ejemplo de una tabla con seis celdas, dos filas y tres columnas:

```
<table>
  <tr>
    <td>celda 1 - fila 1 columna 1</td>
    <td>celda 2 - fila 1 columna 2</td>
    <td>celda 3 - fila 1 columna 3</td>
  </tr>
  <tr>
    <td>celda 4 - fila 2 columna 1</td>
    <td>celda 5 - fila 2 columna 2</td>
    <td>celda 6 - fila 2 columna 3</td>
  </tr>
</table>
```

Ahora, vamos a extender la celda 1 para que abarque la celda 2:

```
<table>
  <tr>
    <td colspan="2">celdas 1+2 - fila 1 columnas 1+2</td>
    <td>celda 3 - fila 1 columna 3</td>
  </tr>
  <tr>
    <td>celda 4 - fila 2 columna 1</td>
    <td>celda 5 - fila 2 columna 2</td>
    <td>celda 6 - fila 2 columna 3</td>
  </tr>
</table>
```

Y también extendemos la celda 4 para que abarque la celdas 5 y 6:

```
<table>
  <tr>
    <td colspan="2">celdas 1+2 - fila 1 columnas 1+2</td>
    <td>celda 3 - fila 1 columna 3</td>
  </tr>
  <tr>
    <td colspan="3">celdas 4+5+6 - fila 2 columnas 1+2+3</td>
  </tr>
</table>
```

Y lo mismo podríamos haber hecho en el otro sentido, extendiendo la celda 1 para que abarcara la celdas 4 hacia abajo:

```
<table>
  <tr>
    <td rowspan="2">celdas 1+4 - filas 1+2 columna 1</td>
    <td>celda 2 - fila 1 columna 2</td>
    <td>celda 3 - fila 1 columna 3</td>
  </tr>
  <tr>
    <td>celda 5 - fila 2 columna 2</td>
    <td>celda 6 - fila 2 columna 3</td>
  </tr>
</table>
```

En las etiquetas `<col>` y `<colgroup>` puede agregarse el atributo `span` para indicar la cantidad de columnas a expandir.

11	12	13	14 + 24 + 34
21 + 22		23	
31	32	33	
41	42	43	44
51 + 52 61 + 62 71 + 72		53	54
		63	64
		73 + 74	

Creación de formularios

Una de las técnicas más utilizadas para que una página web tenga cierto grado de interactividad es el uso de los llamados formularios.

Los formularios permiten que los visitantes de una página transmitan o reciban información específica. Aunque las etiquetas para formularios no son muchas, tienen un número muy amplio de atributos y, algunos de ellos, son muy complejos. En HTML, un formulario es todo lo que se encuentra entre la etiqueta `<form>` y la etiqueta `</form>`; por ejemplo:

```
<form action="URL" method="post" enctype="text/plain">  
  <!-- los elementos para introducir datos y enviarlos -->  
</form>
```

El atributo **method="post"** indica que los datos serán enviados cuando se pulse un botón de envío y que no habrá verificación de ningún tipo. El atributo **enctype="text/plain"** indica cómo serán enviados los datos, en este caso, como texto plano.

Estos son otros atributos del elemento `<form>`:

accept-charset indica la codificación de caracteres utilizada

action es la dirección URL a donde se enviará el formulario

autocomplete indica si el elemento permite que el navegador auto-complete los datos

name es el nombre que le damos al formulario

novalidate indica que el formulario no se verificará antes de ser enviado

onreset es una función de JavaScript que se ejecuta cuando se reinicia el formulario

onsubmit es una función de JavaScript que se ejecuta cuando se envía el formulario

Para introducir datos se utiliza la etiqueta `<input>` que puede ser de muchos tipos; por ejemplo, un cuadro de textos:

```
<input name="minombre" value="" type="text">
```

También es posible introducir un texto y que los caracteres sean irreconocible, por ejemplo, que se vean como los típicos asteriscos que aparecen cuando se nos pide ingresar una contraseña. Para esto, hay que usar el atributo **type="password"**

```
<input name="misecreto" type="password">
```

El elemento esencial en cualquier formulario es el botón de envío que se consigue con el atributo **type="submit"** que puede incluir un opcional llamado **formaction** que indica donde enviar los datos.

EL otro botón que puede agregarse con facilidad es el que permite borrar los datos introducidos antes de enviarlos y es **type="reset"**

```
<input type="submit" value="enviar datos">
<input type="reset" value="borrar datos">
```

El botón de envío puede sustituirse por una imagen utilizando la misma etiqueta pero con **type="image"** y agregando el atributo **src** donde colocaremos la dirección URL de la imagen:

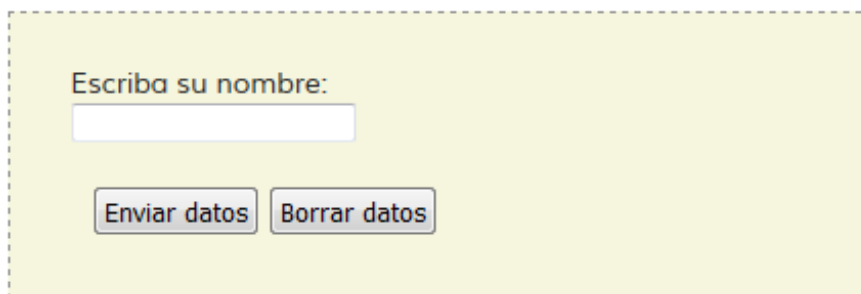
```
<input type="image" src="urlimagen">
```

Así como **type** indica el tipo de elemento, el atributo **value** contiene el valor del dato, por ejemplo un texto por defecto.

Veamos un ejemplo simple, supongamos que deseamos crear una lista de usuarios. Cada uno deberá introducir su nombre y pulsar un botón para enviarnos los datos. Estos datos los recibiremos directamente en nuestro correo, y con ellos confeccionáramos la lista manualmente.

```
<form action="mailto:dirección_de_email" method="post" enctype="text/plain">
  <p>Escriba su nombre</p>
  <input type="text" name="nombre" size="20" maxlength="4">
  <input type="submit" value="enviar datos">
  <input type="reset" value="borrar datos">
</form>
```

La longitud por defecto es de 20 caracteres pero se puede variar incluyendo en la el atributo **size**. Sea cual sea la longitud del formulario, si no se indica nada, puede introducirse cualquier cantidad de caracteres. Se puede limitar incluyendo en la etiqueta el atributo **maxlength** que indica la cantidad máxima de caracteres aceptados

A screenshot of a web form. It has a light yellow background with a dashed border. At the top, it says "Escriba su nombre:" in a dark font. Below this is a white text input field with a thin border. At the bottom, there are two buttons: "Enviar datos" and "Borrar datos", both with a light gray background and a thin border.

Otro atributo que podemos usar es **placeholder** donde podríamos escribir el texto que se mostraría mientras el usuario no ingresara ningún dato y de esa manera eliminaríamos el texto superior:

```
<p><input type="text" name="nombre" size="20" maxlength="4" placeholder="escriba su nombre (hasta 4 caracteres)"></p>
```

Si queremos que el usuario confirme una opción determinada, podemos usar un atributo **type="checkbox"**

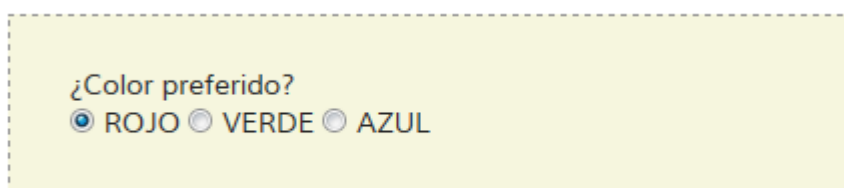
```
<p><input type="checkbox" name="miscondiciones"> aceptar condiciones del servicio</p>
```


Y si queremos que esa opción aparezca marcada por defecto, añadimos el atributo **checked**:

```
<p><input type="checkbox" name="miscondiciones" checked> aceptar condiciones del servicio</p>
```

Cuando queremos que el usuario elija una única opción entre varias, usamos los botones de radio, con **type="radio"** donde usamos el atributo opcional **checked** para que se muestre una de ellas marcada por defecto:

```
<p>¿color preferido?</p>
<p><input type="radio" name="colorpreferido" value="rojo" checked> rojo</p>
<p><input type="radio" name="colorpreferido" value="verde"> verde</p>
<p><input type="radio" name="colorpreferido" value="azul"> azul</p>
```

Una captura de pantalla de un formulario web con un fondo amarillo claro y una bordura punteada. El formulario contiene el texto "¿Color preferido?" seguido de tres opciones de radio: "ROJO" (seleccionada), "VERDE" y "AZUL".

La etiqueta **<label>** se utiliza en los formularios para mostrar textos que sólo son indicativos pero, aunque no tienen ninguna función específica, se pueden asociar con un control agregando el atributo **for** cuyo valor será el de la etiqueta **<input>** correspondiente.

En ese caso, hacer *click* en el texto de **<label>** o en el control **<input>** es indistinto:

```
<p>¿color preferido?</p>
<label for="rojo">rojo</label>
<input type="radio" name="colorpreferido" id="male" rojo="rojo" checked>
<label for="verde">verde</label>
<input type="radio" name="colorpreferido" id="verde" value="verde">
<label for="azul">azul</label>
<input type="radio" name="colorpreferido" id="azul" value="azul">
```

Un tipo de etiqueta **<input>** muy común es el que contiene el atributo **type="button"** que por lo general se utiliza para ejecutar funciones de JavaScript:

```
<input type="button" value="haz click acá" onclick="alert('HOLA');">
```

Otro tipo utilizado con frecuencia es **type="hidden"** que se utiliza para enviar datos extras que no son visibles:

```
<input type="hidden" name="midato" value="esto no se ve">
```

Desde la llegada del HTML5, los tipos disponibles para la etiqueta **<input>** se han ampliado y los navegadores reconocen muchas más que nos permiten filtrar los datos ingresados o seleccionar datos muy específicos.

type="color" permite que seleccionemos un color desde una paleta cuya forma dependerá del navegador que usemos:

```
<input type="color" name="micolor">
```

type="date" se usa para ingresar fechas donde es posible usar los atributos **min** y **max** para limitar los valores:

```
<input type="date" name="mifecha">
```

El ingreso de fechas tiene otras variantes:

type="datetime-local" admite el ingreso de fecha y hora

type="month" admite el ingreso de un mes y un año

type="time" admite el ingreso de una hora

type="week" admite el ingreso de una semana y un año

type="email" se usa para ingresar direcciones de correo:

```
<input type="email" name="miemail">
```

type="file" se usa para cargar algún tipo archivo y allí podemos agregar el atributo **accept** para indicar el tipo o tipos de archivos aceptados:

```
<input type="file" name="miarchivo">
```

type="number" se usa para limitar el ingreso valores numéricos donde **min** y **max** son los valores mínimos y máximos permitidos:

```
<input type="number" name="losdatos" min="1" max="10">
```

type="range" es similar y se utiliza para ingresar datos entre dos valores

type="search" se usa para buscar textos y se combina con el atributo **onsearch** que ejecuta una función de JavaScript.

type="tel" se usa para ingresar números telefónicos

type="url" se usa para ingresar direcciones URL

```
<input type="url" name="miurl">
```

EL HTML5 también ha incluido algunas etiquetas adicionales que se combinan con la etiqueta **<input>**. Por ejemplo **<keygen>** se usa mostrar dos alternativas de encriptación que el usuario puede seleccionar antes de enviar un dato.

Esta clave privada será guardada por el navegador y usando el atributo **keytype** se puede seleccionar el algoritmo a utilizar (rsa, dsa, ec) para generarla:

```
<form>
  nombre: <input type="text" name="usuario">
  seguridad: <keygen name="seguridad">
  <input type="submit">
</form>
```

La etiqueta **<output>** se usa para mostrar el resultado de algún cálculo pero esto no ses algo que funcione de modo automático; copiamos el ejemplo de [MDN](#) y nada más;

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  <input type="number" id="a" value="50">
  +
  <input type="number" id="b" value="50">
  =
  <output name="x" for="a b"></output>
</form>
```

Controles para formularios

Vimos que cuando se quiere que el usuario ingrese un texto, utilizamos la etiqueta `<input>` con el atributo `type="text"` pero, si el texto a introducir es muy largo, es conveniente utilizar un control distinto que admita texto de líneas múltiples y para eso existe la etiqueta `<textarea>`

```
<textarea name="mitexto"></textarea>
```

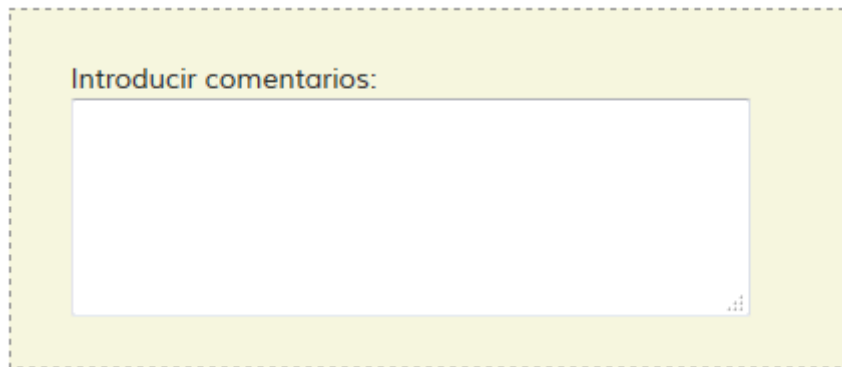
Esta etiqueta admite varios atributos adicionales para delimitar el contenido;

`cols` indica el ancho visible

`rows` indica la cantidad de líneas visibles

`maxlength` indica la cantidad máxima de caracteres aceptados

```
<textarea name="nombre" rows="10" cols="20" maxlength="100"></textarea>
```



Si en lugar de introducir un texto queremos que el usuario escoja entre varias opciones que ya hemos predeterminado podemos usar la etiqueta `<select>` y, dentro ella, agregar la etiqueta `<option>` para mostrar cada una de ellas:

```
<form action="mailto:dirección_de_email" method="post" enctype="text/plain">
  <p>seleccione un color:</p>
  <select name="micolor">
    <option>rojo</option>
    <option>verde</option>
    <option>azul</option>
  </select>
</form>
```

Por defecto, sólo se ve una opción (la primera) pero podemos mostrarla todas añadiendo el atributo `size` que indica la cantidad de filas visibles y, si permitimos que se puedan seleccionar varias opciones al mismo tiempo, el atributo `multiple`:

```
<select name="micolor" multiple size="2">
```

Este tipo de selección también permite elegir entre distintas direcciones web y actuar como vínculos, para esto se agrega el atributo **value** con las direcciones URL:

```
<option value="http://www.google.com.ar/">google</option>
```

Usando el atributo **selected** podemos definir previamente cuál es la opción seleccionada por defecto:

```
<option selected>rojo</option>
```

El atributo **label** se usa en listas desplegables donde las opciones están jerarquizadas y esto suele hacerse agregando una etiqueta extra llamada **<optgroup>** en donde **label** será el texto a mostrar a manera de título:

```
<select>
  <optgroup label="animales">
    <option value="leon">león</option>
    <option value="tigre">tigre</option>
  </optgroup>
  <optgroup label="árboles">
    <option value="pino">pino</option>
    <option value="roble">roble</option>
  </optgroup>
</select>
```

Una alternativa que se ha agregado en HTML5 es que en una etiqueta **<input type="list">** podemos tener agregada una serie de datos que son aquellos que esperamos que el usuario escriba.

Para esto se utiliza la etiqueta **<datalist>** combinada con la etiqueta **<option>** y el atributo **list** de la etiqueta **<input>** debe contener el id de la etiqueta **<datalist>**

Por ejemplo, solicitamos que se ingrese el nombre de un continente:

```
<input list="continentes">
<datalist id="continentes">
  <option value="Asia">
  <option value="Europa">
  <option value="América">
  <option value="Oceanía">
  <option value="Africa">
</datalist>
```

La etiqueta **<datalist>** funciona con un auto-completado así que apenas el usuario ingresa una letra, se despliegan las opciones disponibles asociadas.

Aunque no es usada habitualmente, la etiqueta **<fieldset>** permite agrupar elementos dentro del formulario, enmarcándolos con un borde. Dentro de ella podemos usar la etiqueta **<legend>** para resaltar el título:

```

<form>
  <fieldset>
    <legend>un ejemplo sencillo</legend>
    <p>nombre: <input type="text"></p>
    <p>apellido: <input type="text"></p>
  </fieldset>
</form>

```

No todas las etiquetas de formularios deben estar dentro de una etiqueta **<form>** ni deben ser usadas para enviar o recibir datos desde algún otro servidor. Muchas veces las podemos usar de manera independiente por ejemplo, para ejecutar funciones de JavaScript.

Una alternativa para los botones es la etiqueta **<button>** que nos ofrece más posibilidades gráficas y además, a diferencia de **<input>**, los navegadores no les dan propiedades por defecto así que es mucho mas sencillo personalizarlas:

```

<button type="submit">haga click acá</button>

```

La posibilidad de agregar contenido también hace la diferencia ya que en su interior, no sólo podemos incluir textos simples sino textos formateados o imágenes:

```

<button type="submit">haga <strong>CLICK</strong> acá</button>

```

```

<button type="submit"></button>

```

```

<button name="enviar" onclick="alert('HOLA');">haga click acá</button>

```

El atributo **required** es otra de las nuevas alternativas que se agregan al HTML5 y lo que hace es indicar que cierto campo debe ser “llenado” y de ese modo se evita que un formulario se envíe incompleto.

```
<input type="text" placeholder="ingresar texto" required>
```

El navegador se encargará del resto mostrando un mensaje de advertencia.

Casi todas las etiquetas de formularios comparten atributos comunes a ellas:

autofocus indica que ese control tendrá el foco apenas se cargue la página

disabled indica que ese control está deshabilitado

readonly indica que ese control sólo puede verse pero no cambiarse

tabindex indica la posición en el orden de tabulación de los controles

Insertar archivos

El HTML tiene una serie de etiquetas que nos permiten agregar elementos externos (audio, vídeo, animaciones, etc) siempre y cuando el navegador los pueda mostrar.

Algunos dicen que la mejor forma de incrustar este tipo de archivos es agregar un simple enlace pero claro, no creo que esto conforme a nadie ya que en ese caso, todo dependerá del usuario que haga *click*. Puede aparecer una ventana preguntando si se quiere descargar el archivo o abrirse el reproductor multimedia en el navegador o como programa externo.

¿Y cómo sabe el navegador qué es lo que debe hacerse? Se basa en el llamado *MIME type* (*Multipurpose Internet Mail Extensions*) que son una serie de especificaciones que nos permiten intercambiar todo tipo de archivos a través de internet (texto, audio, vídeo, etc.). Por ejemplo:

```
<a href="URL_archivo.mp3" type="audio/mpeg">un audio</a>  
<a href="URL_archivo.avi" type="vídeo/avi">un vídeo</a>
```

Claro que normalmente pretendemos incrustar el archivo multimedia en la misma página. Para hacer esto, muchos servicios nos proveen un código que copiamos y pegamos. No es una mala solución pero muchas veces no funciona y sobre todo, no nos deja entender qué estamos haciendo de tal manera que, si alguna vez nos encontramos frente a una alternativa, a un formato poco común, no sabemos cómo resolverlo ni dónde buscar la respuesta.

La etiqueta genérica para insertar archivos es `<object></object>` y la sintaxis genérica es la siguiente:

```
<object data="URL_archivo" width="ancho" height="alto" type="MIME type">  
  <param name="nombre" value="valor">  
</object>
```

Hay [MIME type](#) para todos los gustos:

type="application/x-shockwave-flash" para archivos de Flash

type="application/mpeg" para archivos MPG o MP3

type="vídeo/mpeg" para archivos de vídeo MPG

type="text/html" para mostrar páginas web

type="text/plain" para mostrar archivos de texto plano

type="text/richtext" para mostrar archivos de texto formateados

type="application/pdf" para mostrar archivos PDF

type="vídeo/x-ms-asf" para mostrar vídeos ASX

type="application/x-quicktimeplayer" para mostrar vídeos de QuickTime

type="vídeo/x-ms-wmv" para archivos de vídeo WMV

type="audio/x-pn-realaudio" para mostrar archivos de RealPlayer

type="audio/x-midi" para mostrar audio en formato MIDI

type="audio/x-mpegurl" para una lista de reproducción M3U
type="audio/x-aiff" para audio en formato AIFF
type="audio/mpeg" para archivos de audio MP3
type="audio/wav" para archivos de audio WAV
type="video/avi" para vídeos AVI

Incluso las imágenes tienen su propio tipo:

type="image/gif"
type="image/jpeg"
type="image/tiff"
type="image/x-png"
type="image/bmp"
type="image/x-emf"
type="image/x-wmf"

VER REFERENCIAS [Los formatos multimedia]

Otra variante que suele usarse en lugar de **<object>** es la etiqueta **<embed>** aunque esta última no admite archivos de tipo HTML.

Tanto **<object>** como **<embed>** pueden incluir los atributos **height** y **width** para definir su tamaño.

De todos modos, si bien aún se puede ver este tipo de etiquetas (y en algunos casos es inevitable usarlas) agregar contenido multimedia se ha simplificado.

Otra forma muy común de insertar contenido externo en una página web es la etiqueta **<iframe>** que ha vuelto a ser muy utilizada ya que es sencilla.

Antes, podía contener muchos atributos que la hacían confusa pero que han sido depreciados y no deben utilizarse (**frameborder**, **longdesc**, **marginheight**, **marginwidth**, **scrolling**). En estos momentos, sólo necesitamos una obligatoria que es **src** donde pondremos la dirección URL y dos opcionales como **height** y **width** que indican su alto y su ancho algo que también puede hacerse con CSS:

```
<iframe src="URL_archivo" width="500" height="500"></iframe>
```

Las etiquetas audio y vídeo

El HTML5 ha llegado para cambiar la forma de insertar archivos multimedia y simplificarlo, evitando el uso de plugins o software especial ya que son los mismos navegadores los que los reproducen.

Hay dos etiquetas que conforman el nuevo estándar para la reproducción de audio y de vídeo, y que, poco a poco se van abriendo camino a través de los intereses comerciales de muchas empresas.

Las etiquetas en cuestión son `<audio>` y `<video>` y son tan simples de utilizar como esto:

```
<audio src="URL_archivo"></audio>
```

```
<video src="URL_archivo"></video>
```

Ambas utilizan el atributo `src` para indicar la dirección URL del archivo pero, además, poseen algunos atributos opcionales:

autoplay se agrega si queremos que comience a reproducirse apenas se abre la página

controls permite que sean visibles los controles de reproducción (volumen, pausa, etc)

height indica su altura en píxeles (sólo en vídeo)

loop se agrega para que el archivo se vuelva a reproducir cuando termine

muted silencia el audio (sólo en vídeos)

poster indica una imagen de pre-carga (sólo en vídeos)

preload indica si se cargara antes o después que el resto de la página

width indica su ancho en píxeles (sólo en vídeo)

Es aconsejable agregar alguna advertencia que se muestre en los navegadores que no soporten estas etiquetas:

```
<video width="320" height="240" src="URL_archivo" autoplay controls  
poster="URL_imagen">su navegador no soporta la etiqueta vídeo</video>
```

La fuente del audio o vídeo también se puede agregar con etiquetas especiales como `<source>` e incluso, indicar varias de estas para que el navegador utilice la más adecuada al dispositivo empleado:

```
<video controls>  
  <source src="URL_archivo.ogg">  
  <source src="URL_archivo.mp4">  
  su navegador no soporta la etiqueta vídeo  
</video>
```

La etiqueta `<source>` tiene un atributo básico llamado `src` donde agregamos la dirección URL del archivo y otros opcionales como **media** que indica el dispositivo para el cual está optimizado y **type** que indica el tipo MIME.

Por ejemplo:

```
<source src="URL_vídeo.mp4" type='vídeo/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="URL_vídeo.ogg" type='vídeo/ogg; codecs="theora, vorbis"'>
<source src="URL_vídeo.webm" type='vídeo/webm; codecs="vp8, vorbis"'>
```

La etiqueta **<track>** no es muy utilizada pero es una alternativa que permite agregar textos como subtítulos que serían visibles al reproducir el archivo.

```
<video width="320" height="240" controls>
  <source src="UR_archivo.mp4" type="vídeo/mp4">
  <source src="UR_archivo .ogg" type="vídeo/ogg">
  <track src="URL_archivo" kind="subtitles" srclang="es" label="Spanish">
</video>
```

Los atributos aceptados son:

default cuando hay varias opciones indica cuál es la que se usará por defecto

kind indica el tipo (captions, chapters, descriptions, metadata, subtitles)

label es el texto de la etiqueta

srclang indica el idioma en el caso de subtítulos

La etiquetas **<audio>** y **<video>** poseen múltiples atributos que se utilizan para controlar la reproducción mediante funciones JavaScript y estas son algunos de ellos:

oncanplay se ejecuta cuando el archivo esta listo para ser reproducido

oncanplaythrough se ejecuta cuando se puede reproducir sin pausas en la carga

ondurationchange se ejecuta cuando cambia la longitud

onemptied se ejecuta cuando hay una desconexión inesperada

onended se ejecuta cuando se llega al final de la reproducción

onloadeddata se ejecuta cuando se carga el archivo

onloadedmetadata se ejecuta cuando se cargan los metadatos del archivo

onloadstart se ejecuta cuando comienza a cargarse

onpause se ejecuta cuando es pausado

onplay se ejecuta cuando está listo para ser reproducido

onplaying se ejecuta cuando se está reproduciendo

onprogress se ejecuta cuando esta en proceso de carga

onratechange se ejecuta cuando cambia la velocidad de reproducción

onstalled se ejecuta cuando el navegador no puede reproducir el archivo

ontimeupdate se ejecuta cuando se cambia la posición

onvolumechange se ejecuta cuando cambia el volumen

onwaiting se ejecuta cuando la reproducción es pausada

Los scripts

Los *scripts* son pequeños programas que se ejecutan en el navegador. El lenguaje más utilizado es JavaScript pero existen otros.

Como cada navegador tienes rutinas diferentes para hacer las mismas cosas, a veces es necesario tener un conocimiento avanzado del lenguaje para implementarlas pero, poco a poco, los navegadores han ido normalizando y han aparecido librerías que ayudan a simplificar las cosas.

Si bien es cierto que no todos los usuarios tienen navegadores que acepten *scripts*, y, en muchos casos, los desactivan, se dice que no es aconsejable depender de ellas para que la página funcione y lo ideal es tener siempre una alternativa. Pero, la realidad indica que esto no es tan así y hoy por hoy, es un elemento fundamental para cualquier desarrollador.

Agregar *scripts* ya es cosa de todos los días, hay que copiar, pegar y rogar que funcionen los miles de ejemplos que hay en la web y, en general, no hay problemas con eso pero, nunca está de más entender la sintaxis y, si es posible, mejorarla.

Generalmente se insertan en el `<head>` o al final del `<body>` pero, dependiendo de lo que hagan, pueden ser ubicadas en cualquier parte de la página. Para agregar un *script*, basta poner la etiqueta y no es necesario ningún atributo extra:

```
<script>  
    // el código  
</script>
```

Se supone que el atributo **type** debería especificar el lenguaje utilizado pero esto ya no es necesario y sólo es obligatorio si el lenguaje NO es JavaScript:

```
<script type="text/javascript">  
    // el código  
</script>
```

Los scripts también pueden estar en archivos externos y para agregarlos usamos el atributo **src** y para advertir a los navegadores sin soporte de *scripts*, existe la etiqueta `<noscript>` que mostrará un texto alternativo:

```
<script src="URL_archivo"></script>  
<noscript>su navegador no acepta scripts</noscript>
```

En el caso de archivos externos disponemos de otros tres atributos opcionales:

async indica si el código se ejecutará de modo asincrónico

charset indica la codificación de caracteres del archivo

defer indica que el código se ejecutará cuando toda la página este cargada

JavaScript tiene sus bemoles y hay códigos sencillos y códigos complejos, tanto unos como otros nos permiten ahorrar trabajo o hacer cosas que de otra manera serían imposibles de resolver.

No es algo tan difícil como parece a primera vista y sólo requiere un poco de paciencia y conocer algunas instrucciones elementales.

El HTML dinámico, también conocido por las siglas DHTML (*Dynamic HTML*) está basado en una idea de lo más simple: convertir las etiquetas tradicionales del HTML en objetos programables, lo que nos permite poder luego manipularlas a nuestro gusto con JavaScript. De este modo, la página puede ser modificada o tener cierta interacción con los visitantes.

Para cambiar las propiedades de un elemento, es esencial que entendamos que son los eventos.

Un evento (*event handler*) es eso que ocurre cuando hacemos algo (un *click*, escribir algo, mover el ratón, enviar un formulario, tocar una tecla, etc) y hay atributos que permiten acceder a ellos o hacer algo frente a una acción del usuario. Siempre comienzan con **on**.

Un ejemplo simple; al hacer *click* en el enlace se mostrará una ventana de alerta:

```
<a href="#" onclick="alert('HOLA')">click acá</a>
```

Los *event handler* pueden ejecutar varias acciones simultáneamente. Esto se consigue separando cada una con un punto y coma. Por ejemplo acá se mostrará una ventana de alerta y se cambiará la página:

```
<a href="#" onclick="alert('ADIOS'); window.location='URL_pagina';">click acá</a>
```

Otros dos eventos muy utilizados son **onmouseover** y **onmouseout**. El primero se activa cuando se pasa el cursor del ratón sobre un objeto y el segundo cuando se lo quita.

Los estilos

Varias veces hemos dicho que muchos atributos han sido depreciados y se aconseja reemplazarlos por estilos CSS ¿qué significa esto?

Las hojas de estilo permiten controlar la presentación de una página web determinando las propiedades de cada etiqueta (fuentes, márgenes, colores, etc).

CSS (*Cascading Style Sheets*) no es un lenguaje (aunque tiene sus reglas y su sintaxis); sólo es una lista de las propiedades que permiten definir reglas genéricas o particulares.

Esto lo podemos hacer de tres maneras diferentes:

1. enlazando un archivo que contenga con las reglas del estilo
2. listando las reglas dentro de una etiqueta **<style>**
3. añadiendo el atributo **style** en una etiqueta

Para el primer caso utilizamos la etiqueta **<link>** y la agregamos dentro del **<head>**

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

Ese archivo es un simple archivo de texto plano cuya extensión en realidad puede ser cualquier otra.

Para listar las reglas dentro del mismo documento HTML utilizamos la etiqueta **<style>** **</style>** dentro de la cual escribimos las reglas y propiedades. Esta etiqueta suele estar siempre dentro del **<head>** pero, eventualmente, pueden agregarse otras en cualquier parte del **<body>**

El atributo **type** que indica el tipo de medio en que va a ser publicado que normalmente es **text/css**. Por ejemplo:

```
<style type="text/css">
  /* esto es un comentario */
  body {
    background-color: yellow;
    font-family: Arial;
    margin: 0px;
  }
  h1 {
    color: blue;
  }
  h2 {
    background-color: red;
    font-size: 16px;
    color: white;
  }
</style>
```

Allí, cada regla está compuesta de dos partes. El selector (el nombre de la etiqueta, un id o una clase) que en el ejemplo podría ser **h1**; y la segunda es la propiedad y su valor.

En el ejemplo, **h1 {color: blue;}** hará que todas las etiquetas **h1** sean de color azul.

El selector es un vínculo entre el HTML y la hoja de estilo. Todos los elementos de un documento son selectores y cada uno de estos posee un conjunto de propiedades que varían.

Para agregar estilo a una etiqueta concreta, debemos añadirle el atributo **style** que contendrá las reglas:

```
<p>este es un texto normal.</p>
```

```
<p style="margin-left: 30px; color:red">este es un texto en color rojo y con sangría.</p>
```

Todo el contenido del atributo **style** se pone entre comillas y en cada definición, el valor se asigna mediante dos puntos (:) y las distintas propiedades se separan con un punto y coma (;).

Para poder utilizar CSS y sacarle el máximo provecho es indispensable entender cuál es la estructura de un documento HTML.

Podemos imaginarnos que una página web es un conjunto de cajas (bloques) metidas una dentro de la otra, siendo la más grande, la que contiene a todas, el elemento definido con la etiqueta **<html>**.

En este tipo de estructura, donde hay elementos contenidos en otros hay una cierta jerarquía, los elementos padre son los contenedores y los elementos hijos los contenidos.

El problema de eso es que como las etiquetas pueden anidarse, hay que descubrir cual esta dentro de cual para saber cuál es el orden de prioridades y eso es lo que se llama herencia.

Muchas de las propiedades de estilo de los elementos padre son heredadas por los elementos hijo, pero no al revés.

```
<p>
  un texto cualquiera
  <strong>
    un texto en negrita
    <em>un texto en itálica</em>
  </strong>
</p>
```

En ese ejemplo, si definimos **p {color: red;}**, TODO será de color rojo; pero si sólo definimos **strong {color: red;}**, sólo lo que esta dentro de la etiqueta **** será de color rojo.

Y lo complicamos poniendo ambos: **p {color: yellow;}** **strong {color: red;}** y así TODO será de color amarillo pero lo que está dentro de **** será rojo.

Las hojas de estilo requieren un estudio más profundo así que sólo enumeraremos algunas de las propiedades más comunes:

background establece el color o imagen de fondo

background-color establece el color del fondo

border establece el ancho, tipo y color del borde

color establece el color del texto

font-family establece la fuente del texto

font-size establece el tamaño del texto

font-weight establece el espesor de la fuente

height especifica el alto

line-height establece la separación entre líneas

margin define el tamaño los márgenes

padding establece el espacio de relleno

text-align permite alinear los elementos

text-decoration permite remarcar el texto

width especifica el ancho

Todos los métodos pueden aplicarse a la vez en una misma página. Podemos tener un archivo externo genérico, una o varias etiquetas **<style>** y usar el atributo **style** en etiquetas individuales.

Para evitar conflictos entre los distintos métodos, existe un orden de precedencia, es decir, cuál regla prevalecerá sobre la otra si existen definiciones contradictorias.

El orden de precedencia de mayor a menor es el siguiente:

1. los estilos definidos dentro de una etiqueta con el atributo **style**
2. los estilos definidos por la etiqueta **<style>**
3. los estilos contenidos en los archivos externos

VER REFERENCIAS [Tabla de colores]

Semántica y estructura

Casi todas las etiquetas tiene atributos específicos y características gráficas por defecto. Quizás, la más importante de esas características es que algunas etiquetas son elementos de bloque y otras no.

Los elementos en bloque pueden contener elementos en línea y también a otros elementos en bloque. Suelen provocar un salto de línea antes y otro después de los contenidos del elemento. Es lo que ocurre con `<p>` (párrafo), `` (lista ordenada) y `` (objeto de lista).

Los elementos en línea ocupan una o varias líneas del texto de un elemento en bloque. En una misma línea puede haber varios elementos en línea y, en general, no se puede especificar su tamaño pero si sus márgenes, bordes y rellenos. Además, no no pueden contener elementos en bloque. Por ejemplo `` (énfasis) y `` (énfasis fuerte).

Hasta la aparición del HTML5 había dos etiquetas que podrían ser la forma más simple de entender esto. Se trata de las etiquetas `<div>` y ``. Ambas carecen de atributos especiales y “no hacen nada ni muestran nada”

La etiqueta `<div>` es una etiqueta de bloque y la etiqueta `` es una etiqueta *inline*.

¿Par que las usamos? Para todo ya que es la forma de agrupar contenido, separándolo y estableciendo la forma en que se mostrará.

```
<div style="color:red">
  <h2>un título</h2>
  <p>un <span style="background-color:white;">texto</span> cualquiera</p>
</div>
```

```
<div style="border:2px solid green">
  <h2> otro título</h2>
  <p>otro <span style="color:white">texto</span> cualquiera</p>
</div>
```

Si bien estas dos etiquetas siguen siendo usadas a destajo, el HTML5 ha introducido un nuevo concepto, la llamadas etiquetas semánticas o estructurales que en realidad, no difieren de las anteriores en cuanto a funcionamiento sino que están pensadas para indicar de modo explícito las distintas partes de una página web.

Se las denomina semánticas porque su nombre (en inglés) indica o pretende indicar qué hay dentro e ellas

Con ese criterio en mente, decimos la etiqueta `<header>` es un contenedor de encabezados, enlaces de navegación, información importante, etc y como tal, puede haber varios dentro de nuestra página.

```
<header>
  <h1>título principal</h1>
  <p>otros datos</p>
</header>
```

La etiqueta **<hgroup>** se usa para agrupar encabezados (etiquetas h1, h2, h3, h4, h5, h6)

```
<hgroup>
  <h1>título principal</h1>
  <h2>título secundario</h2>
</hgroup>
```

La etiqueta **<main>** es donde encontraremos el contenido principal de nuestra página y es contenido debería ser único, no repetirse dentro de la misma página.

Las etiquetas **<article>** y **<section>** son muy parecidas. La etiqueta **<section>** se usa para marcar una sección, un contenido agrupado temáticamente. La etiqueta **<article>** se usa para marcar contenido independiente (noticias, comentarios) por lo tanto, sólo debe haber una etiqueta **<main>** ni debe estar contenida por otras.

```
<main>
  <h2>título</h2>
  <p>cualquier contenido</p>
  <section>
    <h3>subtítulo</h3>
    <article>
      <header>
        <h2>título</h2>
        <p>autor</p>
      </header>
      <p>... contenido ...</p>
    </article>
    <article>
      <header>
        <h2>título</h2>
        <p>autor</p>
      </header>
      <p>... contenido ...</p>
    </article>
  </section>
</main>
```

La etiqueta **<footer>** indica el pie de una sección o de la página.

La etiqueta **<nav>** se utiliza para marcar un sector con los enlaces de navegación más importantes de la página o sitio.

```
<nav>
  <a href="home">inicio</a> | <a href="contacto">contacto</a>
</nav>
```

La etiqueta **<aside>** se utiliza para marcar un sector relacionado con el contenido de la página pero que es accesorio (notas, referencias, etc).

Las etiquetas **<details>** y **<summary>** se utilizan para crear alguna clase de sección interactiva que el usuario pueda abrir o cerrar y su contenido no debería ser visible sin la intervención de este:

```
<details>
  <summary>detalles a ver</summary>
  <p>cualquier contenido</p>
</details>
```

<details> admite dos atributos:

ontoggle ejecuta una función JavaScript cuando se abre o cierra

open indica que el contenido es visible

No está muy claro si estas etiquetas finales pertenecen al grupo de semánticas o estructurales pero son nuevas en HTML5:

Las etiquetas **<figure>** y **<figcaption>** se utilizan para identificar imágenes y eventualmente, su título o descripción:

```
<figure>
  
  <figcaption>explicación de la imagen</figcaption>
</figure>
```

La etiqueta **<meter>** se utiliza para marcar de modo gráfico cierto dato en determinado rango; por ejemplo:

```
<meter value="5" min="0" max="10">2 out of 10</meter>
```

Donde el atributo **value** es el valor actual y los atributos **min** y **max** indican los valores mínimos y máximos.

Eventualmente, también pueden usarse los atributos **optimum**, **low** y **high** para indicar los valores considerados óptimos, mínimos y máximos.

La etiqueta **<progress>** se utiliza para representar gráficamente el proceso de alguna acción (como el tiempo de carga, etc) donde el atributo **value** indica el valor actual y **max** el máximo.

```
<progress value="5" max="100"></progress>
```

La etiqueta canvas

La etiqueta `<canvas>` también es una nueva etiqueta de HTML5 que nos permite generar gráficos de manera dinámica en una página y sólo posee dos atributos: **height** y **width** que determinan su tamaño aunque si no los definimos, el tamaño por defecto será 300x150 píxeles.

```
<canvas id="migrafico" width="valor" height="valor"></canvas>
```

Es una etiqueta vacía, un contenedor donde puede escribirse, dibujarse o agregarse imágenes que se controlan con JavaScript y por lo tanto, pueden ser modificados en tiempo real, creándose ilustraciones, animaciones, juegos y casi cualquier cosa.

Como es una etiqueta vacía, no veremos absolutamente porque es transparente aunque es posible agregarle algunas propiedades como márgenes, bordes, fondos, etc con CSS.

Lo que hace la etiqueta no es otra cosa que crear un “lienzo”, un espacio particular en donde podremos dibujar pixel por pixel utilizando JavaScript así que, para empezar a manipular su contenido debemos empezar a escribir funciones en ese lenguaje.

Un ejemplo simple.

```
<script>
  var miCANVAS = document.getElementById("migrafico");
  if (miCANVAS.getContext) {
    var canvas = miCANVAS.getContext("2d");
    canvas.fillStyle = "#ffffff";
    canvas.fillRect (0,0,300,200);
  } else {
    // advertencia : el navegador no soporta canvas
  }
</script>
```

Esta son algunas funciones accesibles:

fillStyle() define el color de lo que dibujaremos

fillRect() dibuja un rectángulo lleno

strokeRect() dibuja un rectángulo con borde

clearRect borra un rectángulo

Las limitaciones de la tipografía

Cuando diseñamos una página web disponemos de muchas alternativas, colores, fondos, imágenes pero, siempre hay algo que nos limita un poco o que no termina de verse como nosotros queríamos: la tipografía.

Esto es así por una limitación de internet en si misma ya que las fuentes de los textos no están en la web sino que las tenemos cada uno de nosotros en nuestros dispositivos; los navegadores las buscan allí y si no las encuentran, utilizan las más parecidas o las que encuentren.

Es usual que, por desconocimiento, usemos alguna que nos guste y cuando miramos nuestro sitio en otro dispositivo nos sorprendamos porque se ve mal.

Hay muy pocas fuentes disponibles en cualquier dispositivo o sistema operativo: Arial, Georgia, Courier, Times New Roman, Verdana.

¿Podemos arriesgarnos a usar otras? Sí, pero asumiendo el riesgo de que cierto número de visitantes no las vea ya que, hay fuentes que "normalmente" están instaladas en un alto porcentaje de sistemas operativos así que, es muy probable que sólo unos pocos usuarios no disponga de ellas: Comic Sans, Lucida Grande, Helvetica, Tahoma, Trebuchet.

Quienes saben de estas cosas dicen que Palatino es una fuente disponible en el 97% de Windows y el 79% de las Macs; Impact es una fuente disponible en el 96% de Windows y el 88% de las Macs; Century Gothic es una fuente disponible en el 85% de Windows y el 42% de las Macs; Copperplate es una fuente disponible en el 56% de Windows y el 86% de las Macs, etc, etc, etc.

El núcleo de fuentes básicas para la web tiene su historia que se remonta hasta 1996 Microsoft cuando inicia el proyecto de crear un conjunto de fuentes estándar que pudieran ser usados en internet. Todo terminó uno años después debido a problemas de copyright, negocios varios y peleas entre los diferentes fabricantes de sistemas operativos. Incluso ahora, en Linux, muchos de ellos no están disponibles por defecto y deben ser descargados y agregados como paquetes de terceros.

Otra limitación está dada porque hay muchos caracteres que no ven o se muestran de modo erróneo.

La codificación de caracteres es una opción que vemos en cualquier navegador y, por lo general, lo que dice es Unicode (UTF-8) que es un estándar para mostrar caracteres de diferentes idiomas que no están disponibles en el juego de caracteres ASCII.

Muchos de esos caracteres son símbolos y no pueden escribirse de modo "normal".

Estos códigos empiezan siempre con el signo & y acaban siempre con ; (punto y coma).

Aquí hay algunos ejemplos:

< para < (less than, menor que)

> para > (greater than, mayor que)

& para & (ampersand)

" para " (double quotation)

De una manera similar, existen códigos para escribir letras específicas de distintos idiomas:

á para la á

é para la é

í para la í

ó para la ó

ú para la ú

ñ para la ñ

¿Esto significa que debemos reemplazar esos caracteres por su correspondiente símbolo? No, es muy posible que el navegador muestre los textos correctamente, pero nunca podremos estar seguros que le ocurra lo mismo a todos los que accedan a nuestras páginas.

VER REFERENCIAS [Tabla de códigos Unicode]

Hasta aquí llegamos, ya es el momento de salir a investigar.

Ahora sí, es tiempo de navegar y mirar los códigos fuente de las páginas. Aprender de lo que otros han hecho.

Eventualmente, surgirán preguntas y los errores serán inevitables. Sólo hay una solución, probar, probar, probar, una y otra vez.

Referencias

¿Por qué Word no sirve para crear páginas web?

Es verdad que casi todos en algún momento hemos querido convertir documentos de Word a HTML y el resultado inevitable ha sido bastante decepcionante porque Word, como casi todos los procesadores de texto, no está "pensado" para generar un código más o menos aceptable. Es una función auxiliar, algo precaria y su uso es muy limitado. Emplearlo para crear entradas en un blog no es aconsejable en absoluto.

Hay dos maneras típicas de usarlo y la peor de todas es guardar el archivo y subirlo a internet ya que, no sólo es inadecuada sino, peligrosa porque el documento generado contiene datos anexos tales como el autor, la empresa y estadísticas privadas que serán visibles para cualquiera.

```
14 <o:DocumentProperties>
15 <o:Author> u</o:Author>
16 <o:Template>Normal</o:Template>
17 <o:LastAuthor> u</o:LastAuthor>
18 <o:Revision>1</o:Revision>
19 <o:TotalTime>0</o:TotalTime>
20 <o:Created>2008-04-29T15:57:00Z</o:Created>
21 <o:LastSaved>2008-04-29T15:57:00Z</o:LastSaved>
22 <o:Pages>3</o:Pages>
23 <o:Words>883</o:Words>
24 <o:Characters>4862</o:Characters>
25 <o:Company> , Inc.</o:Company>
26 <o:Lines>40</o:Lines>
27 <o:Paragraphs>11</o:Paragraphs>
28 <o:CharactersWithSpaces>5734</o:CharactersWithSpaces>
29 <o:Version>12.00</o:Version>
30 </o:DocumentProperties>
```

Copiar y pegar es más seguro pero, Word añade etiquetas extras que son inútiles fuera de su entorno y que, en muchos casos, son visibles en nuestras páginas.

Microsoft tiene otro formato de archivo HTML llamado página web filtrada. Con este tipo de formato se guarda un documento sin tanta información adicional pero, sigue teniendo algunas. Por ejemplo, este tipo de etiqueta es bastante normal:

`<p class="MsoNormal">`

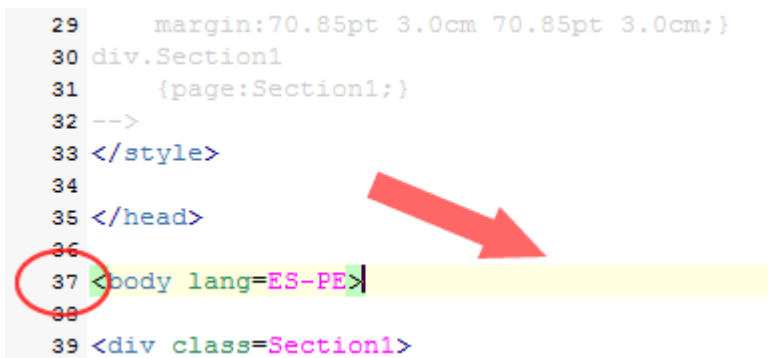
El tamaño del código también se incrementa e incluso, llega a triplicarse lo que significa que la carga de nuestras páginas será más lenta.

```
423 <![endif]--><!--[if gte mso 9]><xml>
424 <o:shapedefaults v:ext="edit" spidmax="205
425 </xml><![endif]--><!--[if gte mso 9]><xml>
426 <o:shapelayout v:ext="edit">
427 <o:idmap v:ext="edit" data="1"/>
428 </o:shapelayout></xml><![endif]-->
429 </head>
430
431 <body lang=ES-PE style='tab-interval:35.4pt
432
433 <div class=Section1>
```

¿Cómo solucionar esto? Lo lógico es no usar Word ni ningún otro procesador de texto.

Si pese a todo, nos gusta Word o nos parece cómodo hay sitios donde podemos convertir esos archivos y "limpiarlos". Por ejemplo, vamos a [Textism](#) y subimos el archivo (hasta 20KB), lo convertimos y luego, copiamos y pegamos el nuevo código.

```
29     margin:70.85pt 3.0cm 70.85pt 3.0cm;}
30 div.Section1
31     {page:Section1;}
32 -->
33 </style>
34
35 </head>
36
37 <body lang=ES-PE>
38
39 <div class=Section1>
```



Los errores más comunes

El que tiene boca se equivoca y el que tiene un quiere hacer algo se equivoca más. Todos nosotros nos equivocamos al escribir códigos, al agregarlos o al corregirlos. Eso no es problema, es lo más natural del mundo. El problema se genera al darnos cuenta del error y al tratar de solucionarlo.

Lo elemental es el diagnóstico y claro, eso es lo más complicado y solemos complicar lo simple hasta que alguien nos dice “el error es ese” y lo miramos como si fuera un genio sin darnos cuenta que, en realidad, estamos tan metidos en nuestro propio trabajo que no llegamos a ver lo que para cualquier otro es evidente.

No, eso no tiene solución, es parte de nuestra naturaleza como seres humanos pero, aún así, hay dos o tres cosas que podemos hacer antes de darnos por vencidos.

Cuando escribimos HTML, el error más común es olvidarnos que existen las etiquetas de cierre. En ciertos servicios esto no es importante ya que se supone (o se suponía) que no todas las etiquetas tienen un cierre ya que algunas se definen por sí mismas: `IMG` `INPUT` `
`, `<meta>`, `<link>`, `<param>`, incluso, si alguien ha aprendido este lenguaje hace mucho tiempo, recordará que se decía que la etiqueta `<p>` tenía un cierre opcional `</p>` y claro, cuando nos dicen que es opcional, solemos optar por lo más cómodo que es no ponerla.

En estos tiempos las cosas han cambiado. No tanto porque ciertos sistemas sigan funcionando igual sino porque algunos tienden a aconsejar que toda etiqueta abierta tenga su cierre.

```

<br/>
```

Pero esto es algo que está cayendo en desuso porque cualquier navegador moderno las interpretará correctamente aunque carezcan de esa barra final.

Que todas las etiquetas estén cerradas tampoco garantiza que no haya un error ya que el orden en que están cerradas también es importante y es otro de los errores más comunes.

La regla de oro es esta: “la última etiqueta abierta es la primera que debe ser cerrada”; por eso, siempre es bueno escribir usando sangrías de tal forma de saber dónde estamos y qué cosas hay abiertas. Esto es un error:

```
<div>
.....
<p> ..... <span> ..... </span>
</div>
.....
<p>
```

Pero como no hemos indentado el código, se nos puede escapar que el `<div>` se cierra cuando la etiqueta `<p>` (que está dentro de este) aún está abierta. Si lo escribimos con sangrías es más sencillo ver cómo debería ser:

```
<div>
  .....
  <p>
    .....
    <span>
      .....
      </span>
    .....
  <p>
</div>
```

También los atributos de las etiquetas tienen sus problemas. El más común es usar el mismo atributo `id` en dos o más etiquetas que, a veces podrá ser inocuo pero debemos evitarlo a toda costa. Los nombres de los `id` son únicos y exclusivos, sólo puede haber uno por página; si queremos repetir propiedades, debemos usar el atributo `class`.

El uso de las comillas debe ser cuidadoso. Al igual que con las etiquetas, hay un orden. Si abrimos una comilla doble y luego otra simple, se deben cerrar en ese mismo orden.

Hay muchos casos en que el problema no es la sintaxis sino que estamos agregando algo que no existe y eso es lo primero que deberíamos chequear. Quizás sea difícil pero el error más común es que estamos colocando una dirección URL errónea.

Esto es así con cualquier archivo externo, *script*, imagen o *favicon* y una forma simple de verificar si son accesibles es copiar la dirección, pegarla en la barra de direcciones del navegador y abrirla. Si es una imagen, un *script*, o un SWF, debería mostrarnos el contenido del archivo. Si es otro formato, debería abrirse o bien descargarse. En ningún caso se abrirá una página web; si ocurre eso es que el servicio que estamos utilizando no admite el uso directo de esos archivos y que se trata de un sitio donde sólo se permite el alojamiento o bien, la URL que estamos usando es incorrecta.

VER REFERENCIAS [Favicons: Ese dibujito que se ve en el navegador]

La etiqueta META

Las etiquetas **<meta>** se utilizan para agregar información general sobre un documento o página web y están descritas por la W3C.

Algunas, sólo son informativas, por ejemplo esta indica el título de nuestro sitio:

```
<meta name="title" content="miPaginaWeb">
```

Y estas el lenguaje y la codificación de caracteres:

```
<meta name="language" content="es">  
<meta http-equiv="content-language" content="Spanish">  
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Estas identifican al autor y el tipo de licencia:

```
<meta name="author" content="mi nombre">  
<meta name="owner" content="mi nombre">  
<meta name="copyright" content="Copyright MIO - Todos los derechos reservados">
```

Esta si el contenido es global, regional o de uso interno:

```
<meta name="distribution" content="global">
```

Esta si el sitio es apropiado o no para menores (**general** | **mature** | **restricted**)

```
<meta name="rating" content="General">
```

Esta si el servicio o programas de diseño que genera el código:

```
<meta name="generator" content="alguno">
```

Pero, lo más común es que escuchemos de ellas en referencia a los motores de búsqueda y a su importancia en la indexación de nuestros sitios. En ese sentido, sólo hay dos etiquetas **<meta>** importantes:

description es un texto mediante el cual describimos nuestro sitio o cada una de sus páginas. Los buscadores leen esta etiqueta y si bien puede ser muy larga, sólo se indexan los 150 primeros caracteres.

```
<meta name="description" content="lorem ipsum dolor sit amet">
```

robots se usa para indicar a los motores de búsqueda si deben o no indexar el contenido de ciertas páginas (**index** | **noindex**) y si queremos que esta acceda a los enlaces que hay en ellas (**follow** | **nofollow**).

Los valores por defecto son **index** y **follow**, así que, en principio, no es necesario indicar nada para que los buscadores indexen el contenido de nuestras páginas y en realidad, se usan para que no lo hagan o, mejor dicho para ciertas páginas no se muestren en los resultados.

Si queremos que se indexe y también sus enlaces (es el valor por defecto):

```
<meta name="robots" content="index,follow">
```

Si queremos que se indexe pero no sus enlaces:

```
<meta name="robots" content="index,nofollow">
```

Si no queremos que se indexe pero si sus enlaces:

```
<meta name="robots" content="noindex,follow">
```

Si no queremos que la página ni su contenido sean indexados;

```
<meta name="robots" content="noindex,nofollow">
```

VER REFERENCIAS [¿follow o nofollow?]

Algunas alternativas de la etiqueta **<meta>** son más sofisticadas:

refresh permite recargar una página en un intervalo de tiempo (segundos) o bien redireccionarla a otra. Esto último debe evitarse porque los buscadores lo consideraran una técnica prohibida y pueden penalizar el sitio, llegando incluso a eliminarlo de sus índices:

```
<meta http-equiv="refresh" content="20">
```

```
<meta http-equiv="refresh" content="20;url=http://otroSitio.com/">
```

expires se usa para indicarle a los buscadores que cierta pagina tiene un tiempo de vigencia y terminado este, debe ser eliminada del índice:

```
<meta meta name="expires" content="never">
```

revisit-after sirve para indicarle a los buscadores que deben volver a visitar la página en cierto tiempo (obvio que no obedecerán y sólo es una sugerencia):

```
<meta name="revisit-after" content="7 days">
```

La caché del navegador también puede ser controlada de tal manera que los visitantes puedan acceder siempre al contenido actualizado. Por ejemplo:

```
<meta http-equiv="Pragma" content="no-cache">
```

```
<meta http-equiv="Cache-Control" content="no-cache">
```

Todas estas definen el idioma y el tipo de caracteres utilizados:

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

Y estas se usan para compatibilidad con móviles y tabletas:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-  
scale=1.0, maximum-scale=1.0, user-scalable=no">  
<meta name="HandheldFriendly" content="True">  
<meta name="MobileOptimized" content="320">  
<meta http-equiv="cleartype" content="on">
```

¿Hay más? Sí. Hay para todos los gustos; incluso los buscadores agregan los suyos. Aquí van las de Google Analytics, Yahoo Site Explorer y MSN Webmasters Tools:

```
<meta name="verify-v1" content="clave_dada_por_el_servicio">  
<meta name="y_key" content="clave_dada_por_el_servicio">  
<meta name="msvalidate.0" content="clave_dada_por_el_servicio">
```

Y algunas que sugiere Facebook que se basan en el protocolo Open Graph:

```
<meta property="og:site_name" content="mi nombre">  
<meta property="og:author" content="mi nombre">  
<meta property="fb:admins" content="ID_cuenta">  
<meta property="fb:app_id" content="ID_aplicacion">  
<meta property="og:url" content="URL_pagina">  
<meta property="og:title" content="nombre pagina">  
<meta property="og:type" content="article">  
<meta property="og:description" content="breve descripción del sitio">  
<meta property="og:image" content="URL_imagen">
```

¿Y entonces? ¿Hay que poner todas esas etiquetas? La respuesta rápida es NO, no es necesario.

Los atributos comunes

Estos atributos son los más comunes y se pueden usar en cualquier etiqueta:

class indica el nombre de una o más clases definidas en la hoja de estilo

id indica un nombre identificador único para un elemento

style enumera propiedades CSS

title permite mostrar información extra del elemento

data-* se utiliza para guardar cualquier información

Estos también se se pueden usar en cualquier etiqueta pero son más específicos:

accesskey indica un atajo de teclado para activar el elemento

contenteditable define si el contenido puede ser editado por el usuario

contextmenu define un menú contextual para el elemento

dir especifica la dirección del texto

draggable define si el elemento puede ser arrastrado

dropzone define dónde se coloca un elemento arrastrado

hidden indica que el elemento no tiene uso

lang define el idioma del contenido

spellcheck indica si el contenido debe ser evaluado por el corrector ortográfico

tabindex indica el orden en la secuencia de teclas de tabulación

translate indica si el contenido debe ser traducido

Todos estos ejecutan alguna clase de función JavaScript y se pueden aplicar a cualquier elemento visible aunque con limitaciones:

onblur se ejecuta cuando el elemento pierde el foco

onchange se ejecuta cuando el elemento cambia

oncontextmenu se ejecuta cuando se abre un menú contextual

onfocus se ejecuta cuando el elemento esta en foco

oninput se ejecuta cuando el elemento es activado

oninvalid se ejecuta cuando el elemento no es válido

onscroll se ejecuta cuando se mueve la barra de desplazamiento del elemento

onselect se ejecuta cuando el elemento es seleccionado

Eventos copiar y pegar:

oncopy se ejecuta cuando el elemento es copiado

oncut se ejecuta cuando el elemento es cortado

onpaste se ejecuta cuando se pega un contenido

Eventos del teclado sobre un elemento:

onkeydown se ejecuta cuando comienza a oprimirse una tecla

onkeypress se ejecuta cuando se oprime una tecla

onkeyup se ejecuta cuando se suelta una tecla oprimida

Eventos del ratón sobre un elemento:

onclick se ejecuta cuando se hace *click* sobre el elemento
ondblclick se ejecuta cuando se hace doble *click* sobre el elemento
onmousedown se ejecuta cuando comienza a oprimirse el botón ratón
onmousemove se ejecuta cuando se mueve el ratón
onmouseout se ejecuta cuando el ratón sale del elemento
onmouseover se ejecuta cuando el ratón se mueve dentro del elemento
onmouseup se ejecuta cuando se suelta el botón del ratón
onmousewheel se ejecuta cuando se usa la rueda del ratón
onwheel se ejecuta cuando se mueve el puntero con la rueda del ratón

Eventos de arrastrar y soltar (*drag&drop*):

ondrag se ejecuta cuando el elemento es arrastrado
ondragend se ejecuta cuando el elemento termina de ser arrastrado
ondragenter se ejecuta cuando el elemento arrastrado entra a una zona válida
ondragleave se ejecuta cuando el elemento es soltado en una zona válida
ondragover se ejecuta cuando el elemento es arrastrado a una zona válida
ondragstart se ejecuta cuando el elemento comienza a ser arrastrado
ondrop se ejecuta cuando el elemento es soltado

Etiquetas y atributos depreciados

Estas son las etiquetas no soportadas por HTML5:

acronym debe usarse <abbr>
applet debe usarse <embed> u <object>
basefont
bgsound debe usarse <audio>
big debe usarse CSS
blink
center debe usarse CSS
dir debe usarse
font debe usarse CSS
frame debe usarse <iframe>
frameset debe usarse <iframe>
hgroup
isindex debe usarse un formulario
listing deben usarse <pre> y <code>
marquee
multicol
nextid
nobr
noembed debe usarse <embed> u <object>
noframes debe usarse <iframe>
plaintext
spacer
strike debe usarse o <s>
tt debe usarse CSS
xmp deben usarse <pre> y <code>

Atributos genéricos depreciados:
align background bgcolor border color

Atributos depreciados en la etiqueta <body>:
alink link text vlink

Atributos depreciados en la etiqueta <object>:
archive classid codebase codetype declare hspace standby vspace

Atributos depreciados en la etiqueta <table> y su contenido:
abbr axis cellpadding cellspacing char charoff frame rules scope nowrap valign width

Atributos depreciados en la etiqueta <iframe>:
frameborder longdesc marginheight marginwidth scrolling

Atributos depreciados en la etiqueta :
hspace longdesc name vspace

Atributos depreciados de las etiquetas **<a>** y **<link>**:
charset coords rev shape target

Atributos depreciados de la etiqueta **<hr>**:
noshade size

Atributos depreciados de la etiqueta **<param>**:
valuetype type

Atributos depreciados de las etiquetas **<dl>**, ****, **<menu>**, **<dl>** y ****:
compact type

Además:
clear en la etiqueta **
**
nohref en la etiqueta **<area>**
profile en la etiqueta **<head>**
scheme en la etiqueta **<meta>**
version en la etiqueta **<html>**
width en la etiqueta **<pre>**

Preguntas y respuestas sobre enlaces

¿Cómo se crea un enlace?

Usando la etiqueta `<a>` (anchor) donde el atributo `href` indicará la dirección URL del documento al que queremos enlazar.

```
<a href="http://misitio.com/">ENLACE</a>
```

¿Qué es una dirección URL?

Una URL (Uniform Resource Locator) es una dirección que permite acceder a un archivo o un archivo (html, php, gif, jpg, mp3, swf, etc). Es la cadena de caracteres que identifica cada recurso disponible en la web:

```
protocolo://máquina/directorio/archivo  
http://misitio.com/index.html
```

¿Se escriben con mayúsculas o minúsculas?

Con minúsculas aunque, dependiendo del sistema operativo, carece de importancia, Windows y Mac ignoran el tipo de letra pero algunos sistemas basados en Unix no.

¿Siempre comienzan con http://?

No. Alguna dirección puede comenzar con `ftp://` o `mailto:`

También podemos usar los llamados *paths* relativos.

```
<a href="archivo.gif">  
<a href="/imagenes/archivo.gif">  
<a href="../imagenes/archivo.gif">
```

¿Todas las direcciones deben terminar con una barra inclinada (slash)?

NO. Esa barra sólo se coloca cuando no se enlazan documentos y colocarla erróneamente, puede confundir al navegador que nos dirigirá a cualquier parte.

La barra final sólo la usamos si no indicamos un archivo y de esa manera dejamos que sea el sitio al que nos dirigimos el que resuelva las cosas:

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

¿Cómo sé si la URL es válida?

El error más común es que digamos "esto no funciona" porque la dirección del recurso que colocamos está equivocada. Si se trata de una página web, basta chequear lo que nosotros escribimos con la dirección que nos muestra el navegador.

Si se trata de otro tipo de archivos debemos hacer lo mismo, copiarla en la barra de direcciones del navegador y hacer que nos muestre el recurso. El resultado, dependerá mucho de que tipo de archivo se trata pero, habrá básicamente dos posibilidades, o nos lo muestra o se abre la ventana de descarga. Si lo que vemos es una página web, la dirección es errónea

¿Cómo hacer que los enlaces abran siempre en una nueva pestaña?

Podemos hacer que se abra en una nueva ventana, que se abra en una nueva pestaña dependerá de la forma en que configuremos el navegador. Para eso, agregamos el atributo **target**:

```
<a href="http://misitio.com/" target="_blank">un enlace</a>
```

¿Podemos abrir ventanas con un tamaño específico?

SI, pero para eso debemos usar JavaScript:

```
<a href="#" target="nombre" onclick="window.open('URL_destino', 'nombre', 'width=600,height=400'); return false">un enlace</a>
```

¿Lo que vemos de un enlace siempre es un texto?

NO. Podemos usar un texto, una imagen o ambos:

```
<a href="URL_archivo"></a>
```

TODO lo que está entre la etiqueta de apertura **<a>** y la etiqueta de cierre **** será el enlace (hipervínculo) y, por defecto, al colocar el ratón encima, veremos un puntero con la forma de una mano

¿Y si la imagen es muy grande, puedo re-dimensionarla?

SI, eso es posible usando los atributos **width** y **height** o bien estableciendo un estilo con CSS:

```
  

```

Pero que se vea pequeña no significa que "se cargará chica". Por ejemplo, si queremos mostrar una imagen de 1024x768, aunque establezcamos que se muestren de un tamaño menor, el navegador cargará la imagen original y luego la re-dimensionará así que el tiempo de carga será incluso superior a si directamente los colocásemos de tamaño natural. En esos casos, conviene tener una segunda imagen ya re-dimensionada.

¿Entonces puede tener una miniatura y enlazar la imagen grande?

SI, eso sería lo razonable:

```
<a href="URL_imagenOriginal"></a>
```

¿Puede quitarse el borde de las imágenes que uso de enlace?

Por defecto, tienen un borde azul. Puede quitarse con CSS:

```
<style>
  a img {text-decoration: none; border: 0; outline: none;}
</style>
```

O, aunque esté depreciado, usar el atributo border:

```
<a href="URL_destino"></a>
```

¿Y el subrayado y color de los textos de los enlaces?

Del mismo modo. Usando CSS:

```
<style>
  a, a:link {color: #color;}
  a:visited {color: #color;}
  a:active {color: #color;}
  a, a:visited, a:link, a:active {text-decoration: none; outline: none;}
  a:hover {color: #color;text-decoration: none; outline: none;}
</style>
```

¿Los enlaces sólo sirven para abrir páginas o recursos?

NO, también los usamos para ejecutar funciones de JavaScript.

```
<a href="#" onclick="mifuncion();">un enlace</a>
```

¿Pueden usarse otras etiquetas para que funcionen como enlaces?

SI, por ejemplo, los botones de los formularios:

```
<form action="URL_destino" method="get">
  <input value="UN ENLACE" type="submit">
</form>
```

¿Cómo creo un enlace para enviar mails?

Usando el protocolo mailto:

```
<a href="mailto:usuario@servicio.com">usuario arroba servicio punto com</a>
```

Cuando se hace *click* sobre ese tipo de vínculo, el navegador abre el cliente de correo por defecto con un nuevo mensaje listo para ser enviado. El atributo **href** puede ser ampliado para que incluya información adicional de la siguiente manera:

`texto`

donde el código adicional puede ser:

cc direccion@servidor.com incluye una dirección en el campo CC

bcc direccion@servidor.com incluye una dirección en el campo BCC

subject texto incluye el texto en el campo ASUNTO

Sólo puede ser utilizada una de las opciones a la vez y, por supuesto, el usuario puede modificar los campos a su gusto antes de enviarlo.

Los formatos de las imágenes

Hay varios motivos para elegir correctamente cuál es el formato de imagen que vamos a usar y, en realidad, no hay reglas en uno u otro sentido, basta tomar la decisión que dependerá de lo que queremos mostrar y de cómo queremos mostrarlo.

Un detalle que no es menor es que los formatos GIF permiten transparencias (que ciertas partes se fundan con el fondo) pero, como contrapartida, la cantidad de colores de un GIF está limitada a 256 y por lo tanto, se pierde cierta definición. Por el contrario, el formato PNG también admite transparencias y mucho más sofisticadas sin limitar la cantidad de colores pero, su tamaño es bastante mayor.

Un ejemplo con la misma imagen guardada en los dos formatos. Como el tamaño es pequeño, no parece haber una gran diferencia pero, eso cambia sustancialmente si la imagen es más grande:

GIF: 9536 bytes 256 colores



PNG: 23109 bytes 6173 colores



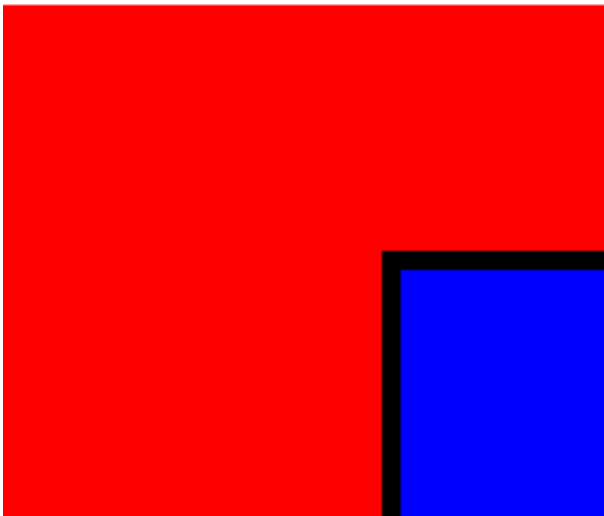
Parece un poco obvio, a mayor calidad, mayor volumen de información y por lo tanto, archivos de mayor tamaño lo que significa tiempos de carga superiores y para peor no hay reglas fijas. El tamaño de ese archivo, no depende exclusivamente del formato con que la guardemos, depende mucho de la imagen en si misma, de la cantidad de colores que tenga, de cómo estén distribuidos, etc.

El formato tradicional de Windows es el BMP y en principio, no tiene compresión ni limitaciones en la cantidad de colores. Son los llamados mapas de bits (BitMaP o Bit Mapped Picture). Para explicarlo rápidamente, cada pixel de la imagen es un número que codifica el color y si tiene más de 256 colores, cada pixel requiere tres números así que una imagen de 250x150 ocupará 112500 bytes más unos cuantos extras que conforman el header o encabezado que lo identifica. Podríamos usar este formato en una página web pero, su volumen es muy importante así que no conviene; sin embargo, esa no-compresión es la que le da la máxima calidad ya que un punto de color jamás cambiará.

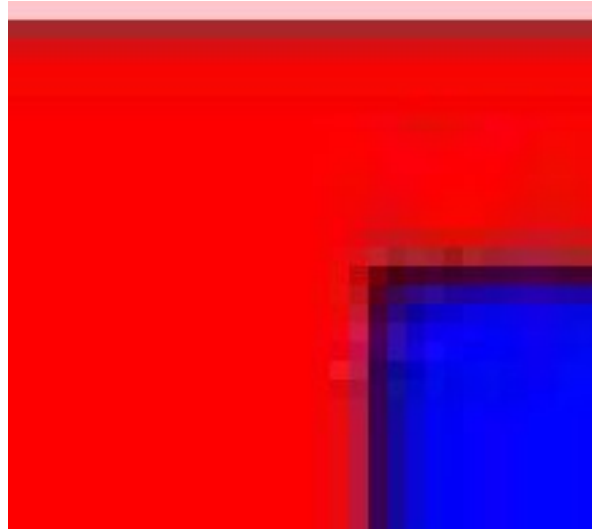
¿Cómo es esto? Es sencillo, la compresión de una imagen hace que "pierda calidad" en ese sentido, la que mayor pérdida tiene es el formato JPG que es el más extendido. Sin duda, es el que genera los archivos más pequeños pero también es el que "deforma" las cosas y eso es muy importante a la hora de elegirlo.

El formato a elegir, dependerá del tipo de imagen a mostrar y no queda otro remedio que probar. Todos tienen sus ventajas y desventajas, lo único que importa es conocer esos beneficios y limitaciones.

Formato PNG



Formato JPG



Las miniaturas que no son miniaturas

Supongamos que escribimos esto:

```

```

¿Estamos cargando una imagen de 128x128?

No necesariamente, o que estamos haciendo es mostrando una imagen con esa dimensión pero estamos cargando la imagen total, sea cual sea su tamaño.

Es un problema de sentido común. Hay un archivo que tiene una imagen, un único archivo. El navegador lo carga y una vez cargado, le decimos que lo muestre de alguna forma y los atributos **width** y **height** o las propiedades CSS equivalentes, sólo hacen eso. Podríamos decirle que lo recorte, que lo coloque acá o allá pero siempre sería ese archivo único y si esa imagen es grande, demorará un poco pero si hacemos lo mismo con varias, las cosas se complicarán ya que el navegador deberá cargar miles o millones de bytes.

Lo mejor, entonces es tomarse el trabajo de recortarlas o re-dimensionarlas previamente dándoles el tamaño que necesitamos.

Esto no significa que siempre debemos usar miniaturas, significa que debemos entender qué estamos haciendo y cómo funcionan las cosas. Cargar una imagen grande y mostrarla pequeña es una técnica aceptable y no tiene nada de malo. Sólo hay que entender qué hacen y usar las herramientas adecuadas.

Pero, si quiero usar **width** y **height** en una miniatura ¿debo calcular su tamaño? ¿cómo puedo hacer para que las imágenes no se deformen?

La respuesta a eso es sencilla y no le gusta a nadie: no se puede. Si queremos que sean cuadradas y la imagen es apaisada, se verá espantosa. Sin embargo, lo que podemos hacer es mantener las proporciones y para esto, basta indicar una sola de sus dimensiones:

```
  

```

El otro valor, el que no definamos, será calculado por el navegador.

Si estamos buscando opciones sencillas, lo ideal es que las imágenes que vamos a utilizar mantengan siempre una proporción adecuada y listo.

Malditas tablas

“Desde que en 1998 las recomendaciones del recientemente creado CSS2 hablaban de ello, el uso de tablas en las páginas web fue cayendo en el olvido y los sitios basados en CSS se transformaron en un sinónimo de elegancia y buen diseño.”

Tablas. Malditas tablas. La cultura geek odia las tablas. Son ... como el Judas Iscariote del HTML. Hay que alejarse de ellas lo más posible. Las tablas, dicen, sólo hay que usarlas para ... hacer tablas ...

Claro, el hecho de que la mayoría de los sitios más visitados usen tablas parece que es un pequeño detalle. Google usa tablas y tan mal no le va. Y no las usa ahora que puede darse el lujo de hacer cualquier cosa, las usó siempre. GMail usa tablas; Yahoo usa tablas, YouTube usa tablas, Twitter usa tablas, casi todos los *gadgets* usan tablas. Parecería entonces que las verdades absolutas y la realidad van por caminos diferentes (como siempre).

Lo elemental sería aclarar algo antes que alguien grite.

No hay etiquetas buenas y etiquetas malas porque no hay códigos "buenos" y códigos "malos". Tampoco hay etiquetas prohibidas porque de existir tal cosa, simplemente, se hubiera eliminado de los navegadores; sin embargo, siguen allí, las usemos o no y algunas ya tienen muchísimos años de existencia.

Pero ¿por qué esos sitios usan tablas?

No sé, tal vez, porque quienes los desarrollaron no saben otra cosa, tal vez porque les resulta cómodo y tal vez, porque a veces es más sencillo. Sea como sea, cualquiera de esas razones o cualquier otra es tan válida como la contraria porque una herramienta es tan buena o mala como quien la utiliza y el diseño web debería estar mucho más allá de estos pseudo-pecados.

Parece obvio pero, en realidad, las páginas web se ven bien o mal, funcionan o no funcionan, se encuentra lo que uno buscaba o no, nos resultan agradables o no y , no importa mucho más porque para eso están.

Filosofía al margen: ¿hay que usar tablas? La respuesta es ... depende, en todo caso ¿para qué usar una tabla? Y bueno, porque a veces, es mucho más sencillo hacer algo de ese modo que de otro y eso, no es ni malo, ni bueno.

Por ejemplo, quiero centrar una imagen tanto horizontal como verticalmente en un rectángulo que tenga un borde y un color de fondo, si sólo es una imagen es sencillo pero si tengo tres imágenes de diferente tamaño y las quiero poner, una al lado de la otra, las cosas se me complican y para centrarlas verticalmente deberé recurrir a alguna clase de truco que implica crear más DIVs, hacer cálculos, usar márgenes negativos, todo eso implica una larga cantidad de CSS. Para colmo, si quisiera cambiar una de ellas, debería empezar otra vez:

Ni lo intento. Obviamente, es mucho más sencillo colocarlas en una tabla donde con un par de atributos, las centro sin problemas:

```
<table style="background:black;border:1px solid red; margin:10px auto;padding:10px;">
  <tr>
    <td></td>
    <td></td>
    <td></td>
  </tr>
</table>
```

Y no, no es el diablito el que me lo sugiere, es el sentido común ¿para qué me voy a complicar la vida? El código usado es válido, es sintácticamente correcto, el resultado es el que quería, funciona, se ve bien ¿puedo pedir más?

¿Esto significa que hay que usar tablas siempre? No, claro que no, ese sería el mismo error que no usarlas. Lo que esto significa es que hay que usarlas cuando nos parezcan útiles.

Tabla de códigos Unicode

































































































▲		▲	▶		►	▼		▼
◀		◄	■		■	□		□
▣		▣	▢		▤	▤		▥
▥		▦	▦		▧	▧		▨
▨		▩	▩		▪	▪		▫
◇	◊	◊	○		○	●		●
☺		☺	☹		☻	☼		☼
Ä	Ä	Ä	Ö	ö	ö	Ö	Ö	Ö
ä	ä	ä	ü	ü	ü	Ü	Ü	Ü
ß	ß	ß	€	€	€	¢	¢	¢
£	£	£	¥	¥	¥	¤	¤	¤
℞		₪	♀		♀	♂		♂
♠	♠	♠	♠		♤	♣	♣	♣
♣		♧	♥	♥	♥	♡		♡
♦	♦	♦	★		★	☆		☆

🏠		⌂	No		№	☎		☎
☎		☏	🔥		♨	👉		☜
👉		☞	🎵		♩	🎵		♪
🎵		♫	🎵		♬	♭		♭
†		†	‡		‡	←	←	←
↑	↑	↑	→	→	→	↓	↓	↓
↔	↔	↔	↕		↕	↖		↖
↗		↗	↘		↘	↙		↙
	 	 	"	"	"	&	&	&
<	<	<	>	>	>	<<	«	«
>>	»	»	©	©	©	®	®	®
™	™	™	¿	¿	¿	¡	¡	¡
@		@	§	§	§	÷	÷	÷
+		+	-		-	Ω		Ω
±	±	±	×	×	×	√		√
1/4	¼	¼	1/2	½	½	3/4	¾	¾

$\frac{1}{3}$		⅓	$\frac{2}{3}$		⅔	$\frac{1}{8}$		⅛
$\frac{3}{8}$		⅜	$\frac{5}{8}$		⅝	$\frac{7}{8}$		⅞
%		%	‰		‰	1	¹	¹
2	²	²	3	³	³			

VER REFERENCIAS [Códigos Unicode extendidos]

Códigos Unicode extendidos

	☀		☁		☂		☃
	☄		★		☆		☇
	☈		☉		☊		☋
	☌		☍		☎		☏
	☐		☑		☒		☓
	☚		☛		☜		☝
	☞		☟		☠		☡
	☢		☣		☤		☥
	☦		☧		☨		☩
	☪		☫		☬		☭
	☮		☯		☰		☱
	☲		☳		☴		☵
	☶		☷		☸		☹
	☺		☻		☼		☽
	☾		☿		♀		♁
	♂		♃		♄		♅
	♆		♇		♈		♉
	♊		♋		♌		♍
	♎		♏		♐		♑
	♒		♓		♔		♕
	♖		♗		♘		♙
	♚		♛		♜		♝
	♞		♟		♠		♡
	♢		♣		♤		♥

◆	♦	♣	♧	☼	♨	♪	♩
♪	♪	♫	♫	♬	♬	♮	♭
♩	♮	♯	♯	✚	♰	✚	♱
✍	✐	➡	✑	✍	✒	✓	✓
✓	✔	✕	✕	✕	✖	✕	✗
✕	✘	✚	✙	✚	✚	✚	✛
✚	✜	✚	✝	✚	✞	✚	✟
✚	✠	✚	✡	✚	✢	✚	✣
✚	✤	✚	✥	✚	✦	✚	✧
♥	❤	☆	✩	★	✪	★	✫
★	✬	★	✭	★	✮	★	✯
★	✰	✱	✱	✱	✲	✱	✳
✱	✴	✱	✵	✱	✶	✱	✷
✱	✸	✱	✹	✱	✺	✱	✻
✱	✼	✱	✽	✱	✾	✱	✿
✱	❀	✱	❁	✱	❂	✱	❃
✱	❄	✱	❅	✱	❆	✱	❇
✱	❈	✱	❉	✱	❊	✱	❋
○	❍	◻	❏	◻	❐	◻	❑
◻	❒	✧	❖	▮	❘	▮	❙
▮	❚	◌	❛	◌	❜	◌	❝
◌	❞	◌	❡	◌	❢	◌	❣
◌	❥	◌	❦	◌	❧	✂	✁
✂	✂	✂	✃	✂	✄	✂	✆
✂	✇	✂	✈	✂	✉	✂	✌
✂	✍	✂	✎	✂	✏		

Favicons: Ese dibujito que se ve en el navegador

El favicon es un ícono; esa pequeña imagen que aparece en los navegadores junto a la dirección URL, ya sea en la barra de títulos del navegador, en las pestañas o en los marcadores.

Se incluyen con etiquetas `<link>` en el `<head>` de la pagina:

```
<link href='URL_imagen' rel='shortcut icon' type='image/x-icon'>  
<link href='URL_imagen' rel='icon' type='image/x-icon'>
```

Ya que algunos navegadores no aceptan cualquier formato de imagen hay que tener cuidado con eso. El estándar es el formato ICO pero puede usarse cualquier otro formato de imagen.

Hay muchos programas gratuitos que permiten crear íconos y sino, se puede recurrir a los servicios online donde podemos subir una imagen cualquiera y ellos la transformarán y nos permitirán descargarla y posteriormente alojarla.

Los estándares web

Los estándares web son palabras que solemos escuchar aquí y allá, todo el tiempo ¿Qué son? ¿Qué ocurrirá si no los sigo? ¿Seré estigmatizado?

Los estándares web son un serie de recomendaciones dadas por el World Wide Web Consortium (W3C) y otras organizaciones que especifican la forma en que deben crearse sitios web ¿Para qué? Esto es lo importante: para que puedan ser accesibles a la mayor cantidad de usuarios posibles. Digo que esto es lo importante porque muchos confunden el fin con el medio. Los estándares y las recomendaciones son un medio, el fin es otro: los usuarios.

El resto es filosofía, algo que discuten los teóricos. Los usuarios eligen porque cierto servicio les sirve, porque les resulta cómodo o porque simplemente, les gusta. Eso es así hasta que dejamos de ser sólo usuarios y pasamos a ser parte de lo que se define como prosumidores, una mezcla de productor de contenidos y consumidor de contenidos. El ejemplo más sencillo de esto es un blog. Ahí, comenzamos a interesarnos en el tema y a usar palabras como estándares, odiar las tablas y tratar de validar nuestro sitio porque ... ¿por qué?

¿Cuáles son las bases de los estándares? Separar el contenido (HTML) de la apariencia (CSS) lo que permite usar menos código y por lo tanto, cargarlos más rápidamente. El estándar, además, permite que los motores de búsqueda identifiquen e indexen los contenidos de los sitios correctamente. Mejora el posicionamiento porque a los buscadores no les gustan los códigos estrafalarios e inútiles. Abarata costos al usar menos ancho de banda, etc, etc.

En todo caso, en la práctica, todo se resume en que el ideal de un sitio es que sea compatible con todos los navegadores y dispositivos, que sea flexible, que sea modificable y además, que sea accesible a personas con discapacidades.

¿Hay alguna objeción a esto? En absoluto, no hay peros ... pero ...

Muchos son los que intentan validar sus sitios leen que hay cientos de errores y se asustan ¿Qué deberían hacer? Una sola cosa, aprender y entender lo que esos mensajes nos están diciendo. Esto es lo básico. Muchos servicios tienen una ventaja, son una forma simple de empezar, pero tienen como consecuencia indeseada, generar en quien lo utiliza, la sensación de que bastan dos o tres *clicks* para resolver las cosas o solucionar problemas. Esto no es así, jamás. Hay algo peor que no saber: creer que sabemos.

Si usamos cualquier herramienta que analice la estructura de nuestro sitio veremos errores que serán imposibles de corregir ya que son provocados por el mismo sistema y por lo tanto, incorregibles y también veremos errores reales que podemos corregir o, por lo menos, saber porque se identifican como errores o advertencias:

"required attribute "alt" not specified" en cada imagen que no posea un atributo **alt**
"required attribute "type" not specified" cuando usamos **<style>** o **<script>**

Habr  que leerlos uno por uno, entenderlos y saber cu l es el l mite entre lo ideal y lo posible.

Pero, por si fuera poco, tambi n veremos errores “dudosos”, esos que nos dicen que algo debe ser escrito de una manera que no parecen tener sentido. Y esto tambi n es importante entenderlo ya que en realidad, el est ndar es un ideal, un punto hacia el que deber amos tender, un camino pero no una regla que deba tomarse sin pensamiento cr tico, evaluando qu  hacer y tomando decisiones subjetivas.

No se trata de infringir las reglas sino de conocerlas ya que de este modo, si las violamos o las eludimos o las cuestionamos, sabremos cuales son las consecuencias y podremos sopesar los pros y los contras de nuestras acciones. No creo en aplicar todas las reglas a ciegas, creo en conocerlas y elegir a conciencia. No nos privamos de hacer ciertas cosas porque hay una ley que lo proh ba, no lo hacemos porque sabemos que est  mal y lo evitamos.

La validaci n es importante pero, debe tomarse como un asistente, sobre todo cuando estamos aprendiendo. Por si misma no significa nada y tampoco implica que se est n usando los est ndares web ya que estos dependen de muchas otras. Un ejemplo:

hola maravilla alegr a sin perros seis avenidas

Esa frase no tiene “errores”, cualquier herramienta que verifique la ortograf a dir  que es correcta y sin embargo, carece de sentido. Valida pero carece de l gica.

Algunos ejemplos pr cticos de la [W3C](#) sobre est ndares:

1. proporcionar a la p gina un buen t tulo que no sea demasiado corto ni demasiado largo:
2. no usar “haz *click* aqu ” como texto de los enlaces ya que si queremos provocar que el usuario haga algo debemos utilizar textos cortos pero significativos, que proporcionen informaci n, que expliquen lo que ofrece el enlace
3. no emplear re-direcciones en etiquetas meta
4. usar el atributo **alt** en las etiquetas **** para proporcionar un texto equivalente; esto, facilita el acceso a personas con problemas visuales y ayuda a los motores de b squeda a indexar correctamente la p gina (los buscadores “leen” ese atributo)
5. si se elige un color, hay que elegir todos. No se puede dejar que sea el navegador qu n defina los colores del texto o del fondo.
6. tener cuidado con el tama o del texto, los tama os peque os est n de moda pero, a n cuando los veamos bien, debemos considerar que a nuestra p gina se acceder  desde otros navegadores, otras plataformas y otros dispositivos as  que no todos ver n lo que nosotros vemos

La meta es siempre la misma, simplificar y clarificar. Nosotros somos los responsables, no las etiquetas ni los navegadores y para eso, no hay ley que valga.

Est ndares SI. Y tambi n experimentaci n, rebeld a, juego, esp ritu cr tico, mente abierta y ser muy consciente de las limitaciones de las herramientas que usamos y de nuestras propias limitaciones. Para todo eso, s lo hacen falta dos cosas: APRENDER y DUDAR.

¿follow o nofollow?

¿Seguir o no seguir? (*That is the question*).

nofollow (no seguir) es un atributo HTML que puede agregarse a los enlaces si queremos que estos no sean tenidos en cuenta por los buscadores. Es un invento diseñado por Matt Cutts (de Google) y Jason Shellen (de Blogger) que comenzó a usarse en el 2005 con el fin de evitar el spam en los comentarios de los blogs.

En poco tiempo, todas las plataformas de blogs y webs importantes, desde Wordpress y Drupal hasta la Wikipedia adoptó el sistema y agregaron el atributo **rel="nofollow"** a sus enlaces con lo que se pretendía evitar que la gente utilizara páginas muy visitadas para conseguir *backlinks* y aumentar su PageRank.

En la práctica, esto:

`enlace a alguna parte`

se transforma en esto:

`enlace a alguna parte`

También puede aplicarse de manera genérica agregando una etiqueta **<meta>** en **<head>**:

`<meta name="robots" content="index, nofollow">`

Google sigue insistiendo en que se siga usando en cualquier lugar donde los usuarios puedan agregar enlaces por sí mismos y no lo hace sólo por evitar el *spam* sino también para que en los resultados de las búsquedas no aparezcan datos irrelevantes.

Algunos especialistas dicen que no es cierto que este tipo de enlace no es tenido en cuenta por los buscadores y hay experimentos realizados que parecen confirmar esa duda.

De un tiempo a esta parte hay una gran discusión en la web sobre el asunto e incluso hay sitios que promueven evitar su uso. La idea se ha extendido y hay muchos blogs que han comenzado a quitarlo de sus páginas.

Los argumentos en uno u otro sentido son variados, quienes recomiendan eliminarlo dicen que su uso no garantiza que nos libremos del *spam*. Dicen: colocar **nofollow** en un enlace es lo mismo que decir, no confío en ese enlace.

Tabla de colores

aliceblue #F0F8FF	antiquewhite #FAEBD7	aqua #00FFFF
aquamarine #7FFFD4	azure #F0FFFF	beige #F5F5DC
bisque #FFE4C4	black #000000	blanchedalmond #FFEBCD
blue #0000FF	blueviolet #8A2BE2	brown #A52A2A
burlywood #DEB887	cadetblue #5F9EA0	chartreuse #7FFF00
chocolate #D2691E	coral #FF7F50	cornflowerblue #6495ED
cornsilk #FFF8DC	crimson #DC143C	cyan #00FFFF
darkblue #00008B	darkcyan #008B8B	darkgoldenrod #B8860B
darkgray #A9A9A9	darkgreen #006400	darkkhaki #BDB76B
darkmagenta #8B008B	darkolivegreen #556B2F	darkorange #FF8C00
darkorchid #9932CC	darkred #8B0000	darksalmon #E9967A
darkseagreen #8FBC8B	darkslateblue #483D8B	darkslategray #2F4F4F
darkturquoise #00CED1	darkviolet #9400D3	deeppink #FF1493
deepskyblue #00BFFF	dimgray #696969	dodgerblue #1E90FF
firebrick #B22222	floralwhite #FFFAF0	forestgreen #228B22
fuchsia #FF00FF	gainsboro #DCDCDC	ghostwhite #F8F8FF
gold #FFD700	goldenrod #DAA520	gray #808080
green #008000	greenyellow #ADFF2F	honeydew #F0FFF0
hotpink #FF69B4	indianred #CD5C5C	indigo #4B0082

ivory #FFFFFF0	khaki #F0E68C	lavender #E6E6FA
lavenderblush #FFF0F5	lawngreen #7CFC00	lemonchiffon #FFFACD
lightblue #ADD8E6	lightcoral #F08080	lightcyan #E0FFFF
lightgoldenrodyellow #FAFAD2	lightgreen #90EE90	lightgrey #D3D3D3
lightpink #FFB6C1	lightsalmon #FFA07A	lightseagreen #20B2AA
lightskyblue #87CEFA	lightslategray #778899	lightsteelblue #B0C4DE
lightyellow #FFFFE0	lime #00FF00	limegreen #32CD32
linen #FAF0E6	magenta #FF00FF	maroon #800000
mediumaquamarine #66CDAA	mediumblue #0000CD	mediumorchid #BA55D3
mediumpurple #9370DB	mediumseagreen #3CB371	mediumslateblue #7B68EE
mediumspringgreen #00FA9A	mediumturquoise #48D1CC	mediumvioletred #C71585
midnightblue #191970	mintcream #F5FFFA	mistyrose #FFE4E1
moccasin #FFE4B5	navajowhite #FFDEAD	navy #000080
oldlace #FDF5E6	olive #808000	olivedrab #6B8E23
orange #FFA500	orangered #FF4500	orchid #DA70D6
palegoldenrod #EEE8AA	palegreen #98FB98	paleturquoise #AFEEEE
palevioletred #DB7093	papayawhip #FFEFD5	peachpuff #FFDAB9
peru #CD853F	pink #FFC0CB	plum #DDA0DD
powderblue #B0E0E6	purple #800080	red #FF0000
rosybrown #BC8F8F	royalblue #4169E1	saddlebrown #8B4513

salmon #FA8072	sandybrown #F4A460	seagreen #2E8B57
seashell #FFF5EE	sienna #A0522D	silver #C0C0C0
skyblue #87CEEB	slateblue #6A5ACD	slategray #708090
snow #FFFAFA	springgreen #00FF7F	steelblue #4682B4
tan #D2B48C	teal #008080	thistle #D8BFD8
tomato #FF6347	turquoise #40E0D0	violet #EE82EE
wheat #F5DEB3	white #FFFFFF	whitesmoke #F5F5F5
yellow #FFFF00	yellowgreen #9ACD32	

Los formatos multimedia

Estos son los formatos de audio más comunes:

- .aac** (Advanced Audio Coding) un formato desarrollado por Apple
- .midi** MIDI (Musical Instrument Digital Interface) son archivos codificados
- .mp3** es el formato más difundido
- .ogg** un formato libre desarrollado por Xiph.Org Foundation
- .rm** y **.ram** (RealAudio) son formatos desarrollados por Real Media
- .wav** un formato desarrollado por IBM y Microsoft
- .wma** (Windows Media Audio) un formato desarrollado por Microsoft

Los únicos formatos soportados por el HTML5 son MP3, WAV, y OGG.

Estos son los formatos de vídeo más comunes:

- .avi** (Audio Video Interleave) es un formato desarrollado por **Microsoft**
- .flv** (Flash) es un formato desarrollado por Macromedia
- .mov** (QuickTime) es un formato desarrollado por Apple
- .mp4** es un formato desarrollado por Moving Pictures Expert Group
- .mpg** y **.mpeg** es un formato desarrollado por Moving Pictures Expert Group
- .ogg** es un formato libre desarrollado por Xiph.Org Foundation
- .rm** y **.ram** (RealVideo) son formatos desarrollados por Real Media
- .swf** (Flash) es un formato desarrollado por Macromedia
- .webm** es un formato libre desarrollado por Mozilla, Opera, Adobe, y Google
- .wmv** (Windows Media Video) es un formato desarrollado por Microsoft

Los únicos formatos soportados por el HTML5 son MP4, WEBM, y OGG.