

Curso básico de HTML 5.

Ya usted, ellos, ellas y yo lo sé: **hay infinidad de tutoriales en internet sobre HTML**: sería "llover sobre mojado" volver a tratar el tema, pero ¿quién dijo miedo? ;-)

En mi caso, el que me ocupa, es actualizarme al [HTML 5](#) y enfrentar a dos viejos "demonios" en esto de la creación de páginas web. Y no, no me refiero a los servicios en GNU/Linux a los cuales llamamos "[D.A.E.MON](#)" si no a los dos grandes problemas que debemos enfrentar antes de comenzar a escribir una sola línea de código en HTML:

1. La multitud de navegadores web existentes y el legado que traemos a rastras, ya que muchas personas se niegan a actualizar sus programas. Agreguemos a eso las particularidades de los nuevos celulares con [Android](#) y ni mencionemos los celulares con [WAP](#) (y por ende [WML](#)) que aún están en uso con sus [micronavegadores](#).
2. Los gustos de nosotros los seres humanos en cuanto a diseño, colores y funcionalidad de las páginas web (en otra entrada presentaré las [hojas de estilo en cascada](#) con las cuales enfrentaremos este aspecto).

Y para no abrumarlos ni abrumarlas más comienzo de una vez el tema.

Prefacio.

La "World Wide Web", tal como la conocemos hoy día, vino de la genial idea del profesor [Tim Berners-Lee](#) (inspirado por [Ted Nelson](#) y convencido por [Richard Matthew Stallman](#)) y es quien actualmente aún lleva las riendas al presidir la "[World Wide Web Consortium](#)", una organización creada en 1994, que agrupa a nivel mundial a más de 300 entes y quienes en conjunto recomiendan las normas del HTML. Y sí, leyeron bien, **RECOMIENDAN** las normativas ya que el HTML no es en realidad un lenguaje de programación sino un [lenguaje de marcado](#), que además descansa sobre dos tecnologías llamada [URL](#) y [HTTP](#). Por ahora sólo necesitamos saber de dónde viene y cuál es la situación actual, si queréis ahondar en conocimientos allí mismo tenéis los enlaces (gracias de nuevo [Señor Tim Berners-Lee](#)).

Los **lenguajes de marcado** son (y para poder explicarlo y que sea comprensible rápidamente) como el antiguo dictado oral que hacíamos a las secretarías y/o mecanógrafas (casi siempre eran mujeres por su talento natural para lidiar con formas, colores y seres humanos) en el siglo pasado:

*"Ponga estilo de carta, ponga comillas, ponga mayúsculas, **Estimado Juan**, ponga dos puntos, aparte, sangría, ponga primera letra mayúscula, **te escribo esta carta**, ponga negrillas, **de forma muy urgente**, cierre negrilla, **ya que no me has enviado...** etcétera".*

Y para los que nacimos y tuvimos la oportunidad de vivir en el siglo pasado, **cada empleada tenía**

su manera de escribir y presentar las cartas, de allí la actuación de los navegadores web, cada uno ofrece ventajas y desventajas y *cada quien escoge de acuerdo a sus necesidades*.

Aclaratoria.

A menos que se diga expresamente lo contrario, todo a lo que nos refiramos es HTML versión 5, la del año 2014, la recomendada. *Por supuesto* que las **etiquetas html** presentan cierta compatibilidad hacia atrás, y también hay unas totalmente nuevas para esta versión. No entraré demasiado en dichas compatibilidades, si queréis estudiar ese aspecto por favor seguid los enlaces que presento a lo largo y ancho de la entrada. *Si ya sabéis, habéis adquirido conocimiento con antelación, os recomiendo un tema más avanzado escrito en este mismo blog:* "[Un CAPTCHA fácil y sencillo de implementar](#)".

Otra cosa que os parecerá harto extraño (pasado el tiempo lo comprenderéis con la experiencia que logréis) yo seguiré la siguiente consigna:

"Cualquiera puede equivocarse, **excepto nosotros**".

¿Qué significa lo anterior? Que seremos más estrictos en la sintaxis del HTML pero sin llegar a declararnos explícitamente como [XHTML](#), así tendremos lo mejor de ambos mundos.

Mi primera página web.

Así que llegamos a dar nuestros pininos en el internet como protagonistas y no como simples lectores. Las páginas web descansan en [archivos de texto plano](#) con la extensión [.htm](#) o para los sistemas operativos modernos [.html](#) y dichas extensiones permiten que identifiquemos qué tipo de contenido albergan más sin embargo para nuestras computadoras el contenido es lo importante. Es así que debemos conocer nuestro primer elemento del lenguaje HTML dentro del archivo nombrado (***recordemos que los nombres de archivos son sensibles a mayúsculas y minúsculas***):

Como vemos comienza con el signo "mayor que" y finaliza con "menor que". *He aquí la primera diferencia con los lenguajes de programación:* aquí dichos símbolos carecen de [significado matemático](#) alguno, sirven más bien como delimitadores. Ya que le damos ese uso, nuestra mente

cambia el enfoque y los asocia a los [paréntesis](#), que trabajan por pares donde uno "abre" y el otro "cierra". Dentro de esta categoría están los corchetes y luego las llaves: ([{ }]), **en ese orden anidado**. Como son rectos y se parecen más a los corchetes que a los otros dos optamos por llamarlos **corchetes angulares** y así son llamados en inglés [angle brackets](#) (y en inglés antiguo **chevrons**). Y [hemos de referirnos](#) al inglés porque precisamente es éste el idioma en el que está escrito el HTML, **angle brackets** es la traducción [literal](#) de **corchetes angulares**.

Dentro de estos delimitadores está el cierre del signo de admiración acompañado de la palabra [doctype](#) que es un recurso nemotécnico abreviado de "DOCument TYPE" y traduciremos como "tipo de documento" acompañado, por supuesto, de la palabra **html**. Para este caso particular el uso de mayúsculas y minúsculas es indistinto pero tiene una excepción: como esta declaración se incluye por compatibilidad con anteriores versiones del lenguaje y dichos identificadores para cada versión son un tanto largas y complicadas *y de paso eran obligatorias* se decidió incorporar lo que yo llamo "un relleno" **que debe ser escrito literalmente y entre comillas dobles o simples para aquellos servidores que buscan estos identificadores**: dicha sentencia es "**about:legacy-compat**" que viene a ser leída por nosotros como "acerca de:compatibilidad heredada". Nuestra primera línea quedaría de la siguiente manera:

La segunda línea a continuación define la página web en sí y tiene el propio nombre **html** para declarar dónde comienza y dónde termina:

...

Donde están los puntos suspensivos colocaremos todo el código, notemos la nomenclatura para definir el idioma y región de nuestro país según el organismo Naciones Unidas: "[es-419](#)"; (y ratificado por el [W3C en esta página](#), sin embargo muchos utilizan la norma [ISO 639-1](#) *ambos son válidos si recordamos que la W3C sólo recomienda, no obliga*) y si necesitáramos programar para alguna otra región del mundo [acá está el resto de los códigos](#).

A partir de este momento colocaré el código completo de la página web las cual ustedes introducirán con el editor de texto de su preferencia, hay muchos, a mi me gusta el [nano](#), [notepad++](#) y [gedit](#). Para uso avanzado están [vi](#) y [vim](#). Las capturas de pantalla que verán son tomadas al **gedit** que viene con **ubuntu** al cual le hice unas cuantas modificaciones de perfil, para verlas en detalle haga click aquí.

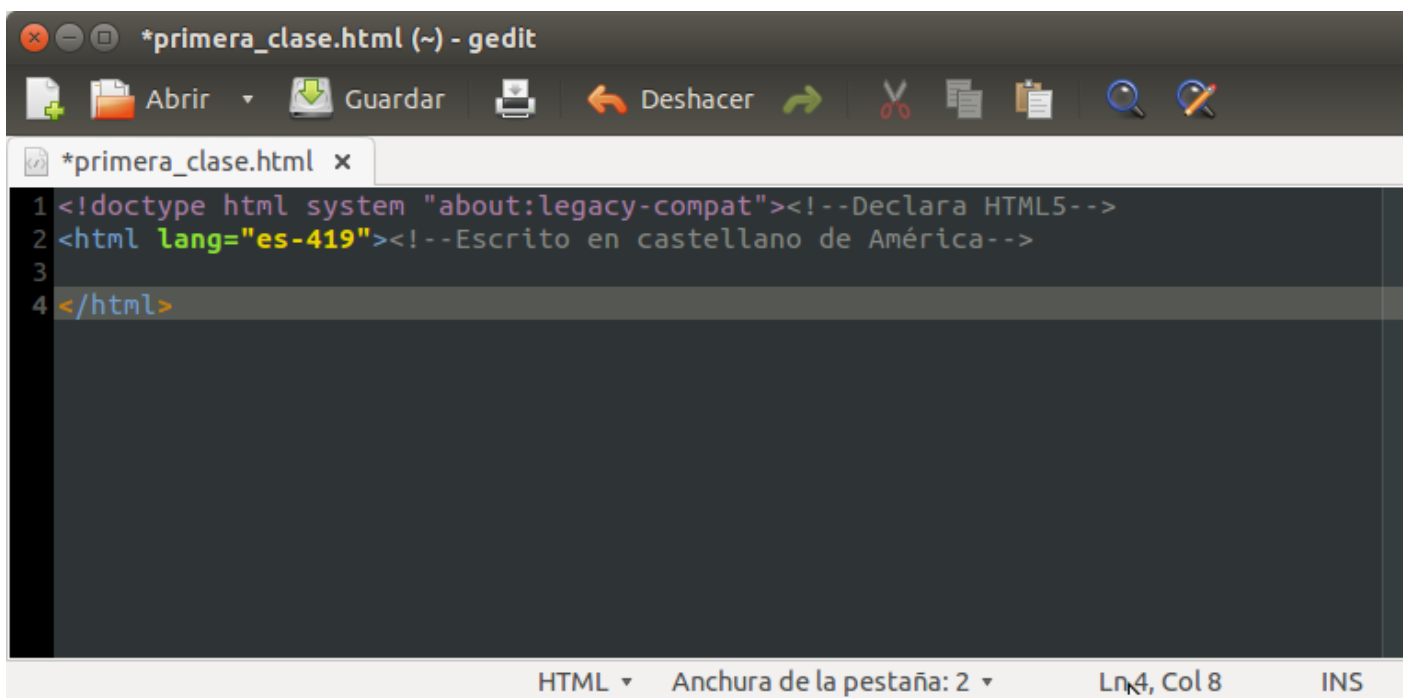
KS7000+WP

KS7000 migra a GNU/Linux y escoge a WordPress para registrar el camino.

<https://www.ks7000.net.ve>

Nosotros los seres humanos necesitamos, nos gusta, tener información adicional que nos guía o aclara el código pero que para las máquinas no tiene relevancia. La manera de que ellas "ignoren" dichos datos es colocándolo como comentarios con la siguiente sintaxis:

Donde están los tres puntos insertamos el comentario que deseemos, no importa si ocupa varias líneas *pero lo importante es que tengan su apertura y cierre*. ¿Recuerdan que les dije que **gedit** lo utilizo con una configuración especial? Comparen el código que llevamos en purito texto plano y vean como lo presenta el editor de texto **gedit**:



The screenshot shows the gedit text editor window titled '*primera_clase.html (~) - gedit'. The menu bar includes 'Abrir', 'Guardar', 'Deshacer', and other standard editing icons. The main editing area shows the following HTML code:

```
1 <!doctype html system "about:legacy-compat"><!--Declara HTML5-->
2 <html lang="es-419"><!--Escrito en castellano de América-->
3
4 </html>
```

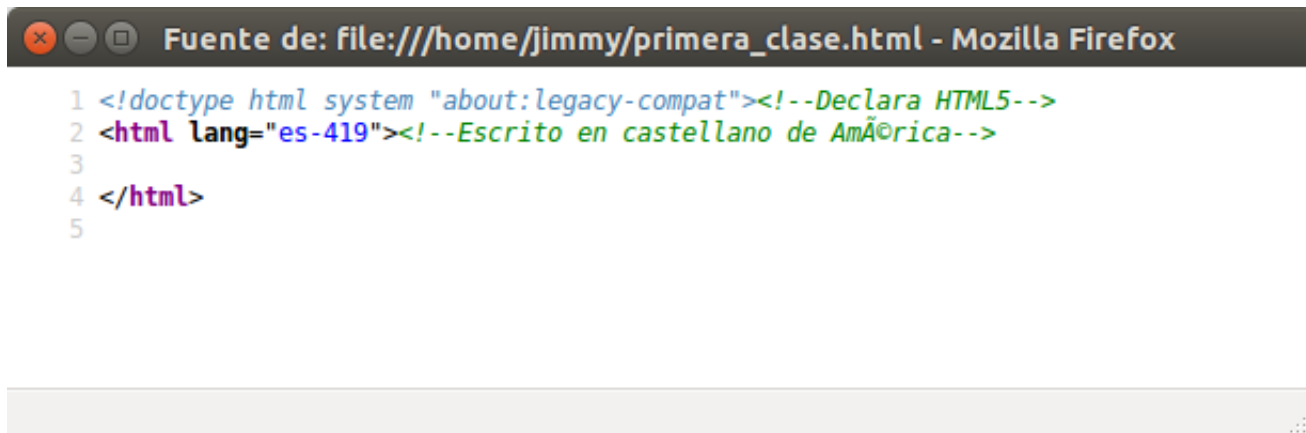
The status bar at the bottom indicates 'HTML', 'Anchura de la pestaña: 2', 'Ln4, Col 8', and 'INS'.

¡Qué diferencia! Al suministrar el nombre al archivo, con todo y extensión, **gedit** automáticamente codifica al lenguaje **html** como pueden observar en la barra de estado inferior de la ventana.

El siguiente paso que haremos es ir a la carpeta donde guardamos nuestro archivo (al que le dimos el nombre "primera_clase.html") y lo abrimos ya sea con doble click, click izquierdo o click derecho+abrir en navegador. Cada sistema operativo moderno tiene un navegador web predeterminado, así es que se puede visualizar nuestra primera página web, **MÁS SIN EMBARGO**

NO VEREMOS NADA PUES AÚN FALTAN MÁS LÍNEAS DE CÓDIGO. ¿Por qué entonces yo pido abrirla de una vez, sin más?

La respuesta es: pedirle al navegador web que nos muestre la página web que estamos haciendo para visualizar el código fuente (generalmente haciendo click derecho sobre nuestra página web abierta en navegador). *Quiero hacer énfasis que una página web, a pesar de todo nuestro empeño de escribirla bien -según las recomendaciones- depende de algunas cosas más.* Les muestro lo que interpreta nuestro navegador web, en este caso [Mozilla Firefox](#):



```
Fuente de: file:///home/jimmy/primer_a_clase.html - Mozilla Firefox
1 <!doctype html system "about:legacy-compat"><!--Declara HTML5-->
2 <html lang="es-419"><!--Escrito en castellano de Am rica-->
3
4 </html>
5
```

El punto que quiero recalcar es la palabra **Am rica** la cual se muestra incorrectamente, con unos caracteres extra os, as  que hemos de reparar en que nuestro bello idioma castellano sea escrito (y visualizado) correctamente. A pesar de derivar del **lat n**, ([idioma que nunca tuvo acentos](#) ni [signos ortog ficos](#)) con la evoluci n de nuestro idioma se lleg  a un *lenguaje de marcado* (ver *inicio de esta entrada*) que ayuda much simo a transmitir mejor las ideas; o lo que yo llamo "[degustar](#) las palabras". Es por ello que en el lenguaje HTML se tom  la previsi n de mostrar correctamente dichos caracteres con etiquetas especiales, as  tenemos que para mostrar nuestras vocales acentuadas y la letra e e (las may sculas siguen la misma normativa):

  -> ´   -> é   -> í   -> ó   -> ú
  -> ñ

Y para los acentos en idioma [catal n](#) y [franc s](#) (aqu  pode s ver una lista de [los m s utilizados](#)):

  -> à   -> è   -> ì   -> ò   -> ù

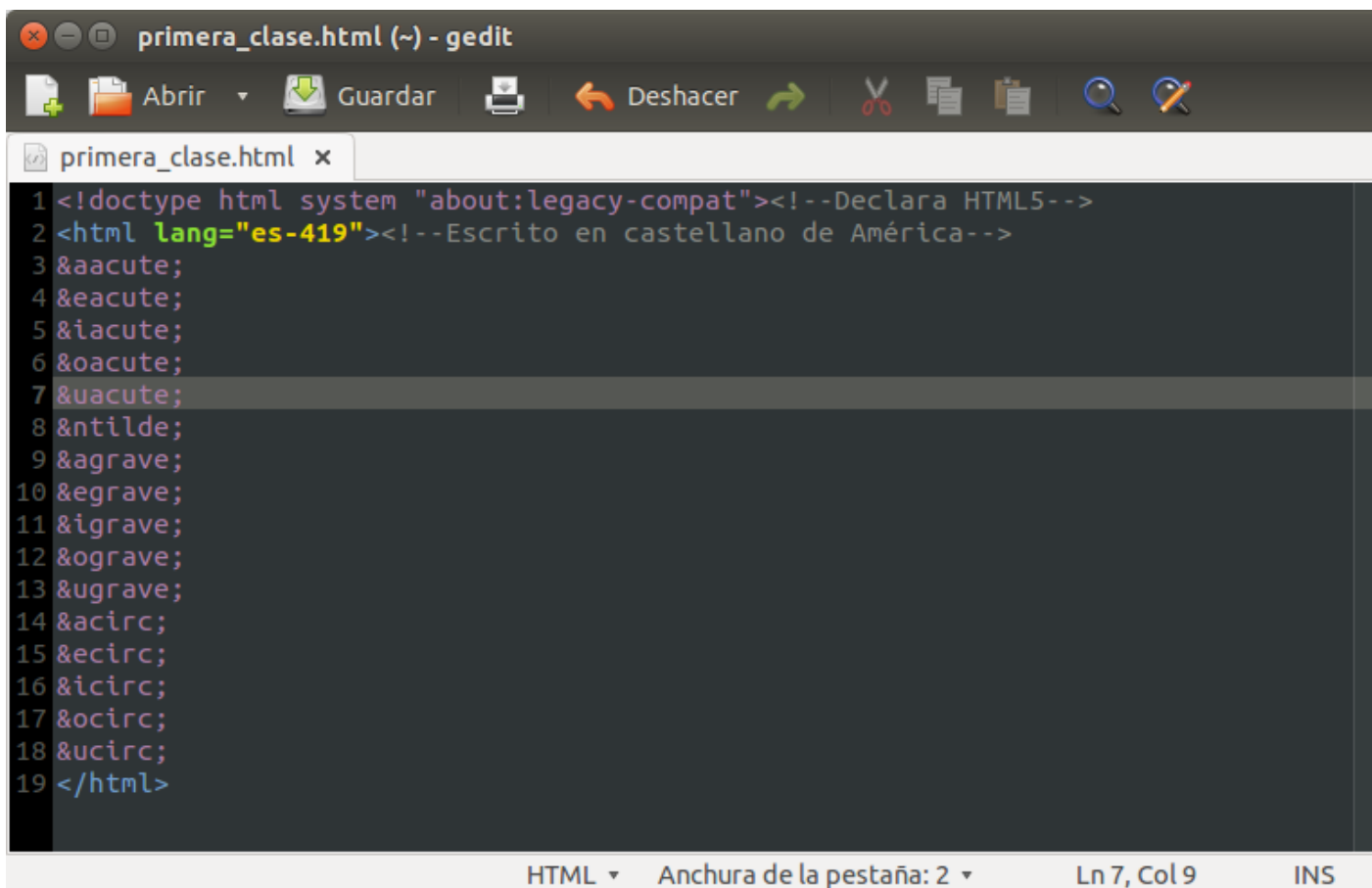
  -> â   -> ê   -> î   -> ô   -> û

KS7000+WP

KS7000 migra a GNU/Linux y escoge a WordPress para registrar el camino.

<https://www.ks7000.net.ve>

Si insertamos los anteriores caracteres "especiales" (no usados en el idioma inglés) en nuestra primera página web, la guardamos y abrimos con nuestro navegador web (y pedimos visualizar nuestro código fuente) veríamos algo similar a esto en el programa **gedit**:



The screenshot shows the gedit text editor window titled "primera_clase.html (~) - gedit". The menu bar includes "Abrir", "Guardar", "Deshacer", and other standard editing icons. The main text area contains the following HTML code:

```
1 <!doctype html system "about:legacy-compat"><!--Declara HTML5-->
2 <html lang="es-419"><!--Escrito en castellano de América-->
3 &aacute;
4 &eacute;
5 &iacute;
6 &oacute;
7 &uacute;
8 &ntilde;
9 &agrave;
10 &egrave;
11 &igrave;
12 &ograve;
13 &ugrave;
14 &acirc;
15 &ecirc;
16 &icirc;
17 &ocirc;
18 &ucirc;
19 </html>
```

The status bar at the bottom indicates "HTML", "Anchura de la pestaña: 2", "Ln 7, Col 9", and "INS".

Y en el navegador web **Mozilla Firefox**:

KS7000+WP

KS7000 migra a GNU/Linux y escoge a WordPress para registrar el camino.

<https://www.ks7000.net.ve>

Como pueden notar el navegador cumple con su trabajo de mostrar los caracteres especiales, más sin embargo podemos observar que la palabra **América** en el comentario sigue sin visualizarse correctamente **y no, no es porque no utilicé el &ecute;** (si quieren hagan la prueba) es por otro motivo más importante. Y, yo ya se que los comentarios no revisten importancia, pensarán ustedes, total eso no se muestra en nuestra página web, *sigamos adelante y comprenderán.*

Mi primera página web es en castellano.

Ya que hay muchísimas páginas web en inglés, pues vaya que le toca al castellano llenar espacio. Antes de proseguir debo aclarar que lo que los voy a comentar lo aprendí [en esta página web](#). (si no está en línea, [prueben este otro enlace](#) o [este otro](#)). Si quieren saber la historia de la

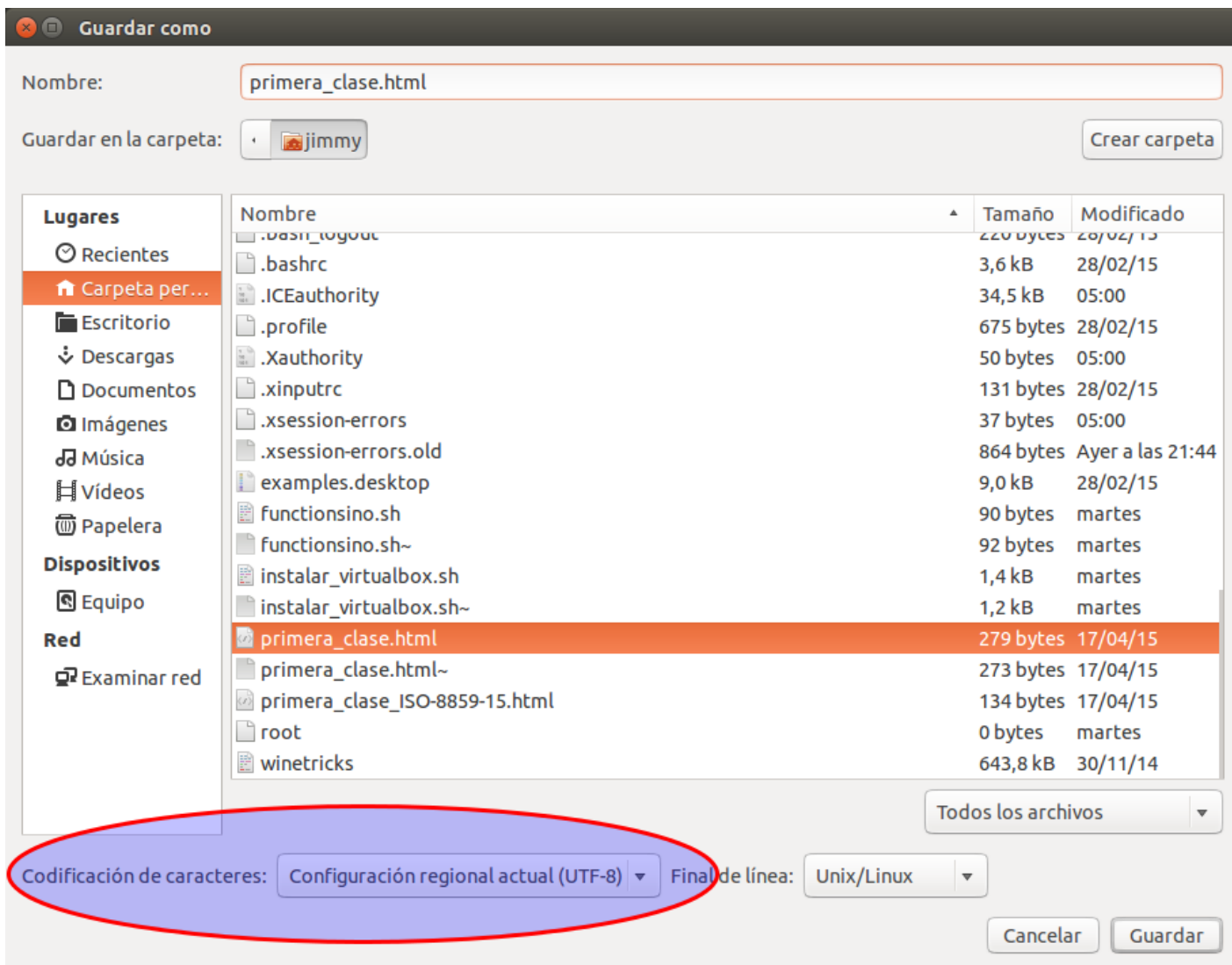
KS7000+WP

KS7000 migra a GNU/Linux y escoge a WordPress para registrar el camino.

<https://www.ks7000.net.ve>

codificación de caracteres puede leer (en inglés) esta otra [página web del lenguaje de programación Python](#). **Debemos indicar también que tocaremos brevemente ciertos aspectos inherentes a los programas que sirven páginas web, "servidores web".**

Volviendo a nuestro archivo de texto plano con extensión .html y editado con el programa **gedit** debemos tener especial cuidado en guardar dicho archivo en la codificación de [caracteres utf-8](#) tal como muestro al escoger la opción "Guardar como", les resalto con una elipse para acentuar la opción:



¿Por qué usar UTF-8 en vez de otros sistemas de codificación?

La respuesta les va a sorprender (y el que me quiera refutar, por favor escriba abajo en los comentarios): **USAMOS EL UTF-8 POR COSTUMBRE HEREDADA DEL [IMPERIO ROMANO](#),**

nacido además en Grecia, cuna de nuestra civilización occidental. Es así, de hecho, que en marzo de 2015 el 83% de las páginas web utilizan esta codificación, y de paso es la recomendada por la [World Wide Web Consortium \(W3C\)](#) por su compatibilidad hacia atrás con la codificación ASCII y hacia adelante con los caracteres (o más bien ideogramas) orientales (hindi, coreano, chino, etc). Esta "compatibilidad hacia adelante" se ve sacrificada por el aumento de bytes en página ya que el **UTF-8** fue pensado para ser de longitud variable: si escribimos en cualquier [lengua indoeuropea](#) usaremos menos espacio y *si escribimos en ideogramas usaremos un poco más de espacio*. Es por ello que elegimos **UTF-8** PERO NO SE LIMITEN A ELLO si se les presenta la oportunidad de trabajar en otro país de Asia (ejemplo loable cuenta Twitter: [@Kopepasah](#)), si ese fuera su caso, están las codificaciones **UTF-16** y la más completa por ahora **UTF-32** ([vean aquí cómo](#) se estructuran los caracteres); agregarlas a nuestro programa gedit es bien fácil, sólo hacemos click en la 'flechita' de codificación de caracteres (ver gráfico anterior) y escogemos "añadir o quitar" y nos muestra este cuadro de diálogo:

Seleccionamos la codificación que necesitaramos y listo, lo guardamos en esa codificación **PERO CON EL CUIDADO ESPECIAL DE ESPECIFICARLO EN NUESTRA PÁGINA WEB.**

El metacomando para declarar la codificación de caracteres escogida es:

y debe ser incluida en el meta comando **head**:

Siendo así que este comando esté declarado en por defecto el penúltimo enlace de ejemplo será buscado en el dominio "**ks7000.net**" en la carpeta "**images**" en vez de la carpeta donde está alojada la página abierta.

Otra opción muy útil es especificar si el enlace se abrirá en la misma ventana o pestaña de donde se le llama o en una nueva, aquí en este tutorial las he colocado en una nueva ventana o pestaña para poder seguir el hilo del tema principal de estudio. Así el comando que necesitamos es:

El otro valores para **target** es "**_self**", que es el valor por defecto y generalmente no se escribe ("**_parent**", "**_top**" y "**framename**", que se usaban con **frame** -no soportado por HTML5- dejaron de usarse). El comando también acepta ambos parámetros.

Imágenes.

La inclusión de imágenes en una página web pasa por hablar brevemente sobre la historias de los formatos utilizados para representarlas. Pero primero veamos el comando utilizado por HTML para decirle al navegador cual imagen debe "cargar" y luego ahondaremos el tema de la historia de las imágenes digitales:

Como ven el comando no tiene un cierre sino más bien un autocierre (opcional en HTML pero obligatorio en XHTML). En este ejemplo asume que la imagen está alojada en la misma carpeta que contiene el archivo .html (a menos que usemos el comando , visto anteriormente dentro de). Si la imagen estuviera ubicada en una subcarpeta, simplemente colocaríamos:

KS7000+WP

KS7000 migra a GNU/Linux y escoge a WordPress para registrar el camino.

<https://www.ks7000.net.ve>

O incluso, si estuviera ubicada en otro servidor de nuestra propiedad (por favor eviten el [hotlinking](#) ya que es de muy mal gusto "robar" el ancho de banda a otros entes o personas):

El parámetro **alt** es obligatorio ya que el navegador muestra el texto colocado entre comillas mientras descarga la imagen en sí y si no la puede descargar, pues bueno, el texto indica que allí debería ir una imagen. Otro uso importante es para los discapacitados visuales ya que los programas lectores usan este texto para narrar de que imagen se trata, así que ojito con colocarle acertadas palabras.

Un uso muy común para las imágenes es utilizarlas como botones para los enlaces ya que llaman más la atención que un mero texto subrayado, aquí debemos aplicar el anidado de instrucciones, fíjense:

Otro parámetro que soporta el comando

es con el cual podemos especificarle al navegador web del usuario que la imagen deseada la convierta a un tamaño específico, ancho y alto en pixeles, por ejemplo:

Así, no importa el tamaño que tenga la imagen original, ésta será mostrada a 128x128 [pixeles](#) pero debe usarse esto con mucho cuidado: **tengan presente que la imagen que descargará el usuario siempre ocupará el mismo ancho de banda, NO IMPORTA QUE SE VEA MÁS PEQUEÑA, la cantidad de bytes descargados siempre será la misma cantidad.** Es por ello que ahora si hablaré de la historia de las imágenes digitales, tómense un café y vuelvan que esto se pone interesante. 8-)

style="text-align: justify;">Imágenes digitales.

En el siglo pasado, cuando usamos rollos fotográficos (plásticos impregnados de químicos fotosensibles para fijar la luz, generalmente hechos a base de [plata](#)) quedaban como [negativos fotográficos](#) con los cuales se podían sacar las fotografías en sí, cuantas copias se necesitaran. Con las fotografías digitales sucede un proceso parecido: cada cámara o filmadora almacena la foto original en formato RAW o DCRAW que, esencialmente, consiste en millones de ceros y unos que representan los colores que recibió el sensor digital. Es, por tanto, muy grande y *tiene muy poca pérdida de información sobre la fotografía tomada*. Pero al igual que con los negativos fotográficos, esta información poco nos sirve por su dificultad de visualizarlos con el ojo humano. Es por ello que, para ahorrar en hardware, se decidió someter dichos datos RAW (existen cientos de formatos RAW, cada fabricante usa o inventa el que mejor le parece **y la mejor alternativa a futuro es el formato libre [DCRAW](#)**) a un proceso de pérdida de información y además compresión y codificación en un formato estandar para, de esta manera, poder compartir con el resto del mundo la fotografía tomada.

Es decir, nuestros navegadores web deben ser capaces de mostrar cualquier imagen siempre y cuando dicho fichero esté codificado en un formato normalizado, el formato RAW o DCRAW no nos sirve para este propósito de hacer páginas web. **Simplificando: un programador de paginas web debe rápidamente hacerse estas preguntas antes de escoger un formato de imagen para el trabajo por el cual lo contrataron:**

- ¿Necesita transparencia?
- ¿Necesita animación?
- ¿Necesita buena calidad de imagen?

Lo que paso a enseñar es un resumen al máximo de lo publicado (en idioma inglés) de las siguientes páginas:

- "[Image Compression for Web Developers](#)" publicado por [Colt McAnlis](#) el 17 de septiembre de 2013.
- "[A Picture Costs A Thousand Words](#)" publicado por [guypod](#) el 19 de junio de 2013.
- "[High DPI Images for Variable Pixel Densities](#)" publicado por [Boris](#)

KS7000+WP

KS7000 migra a GNU/Linux y escoge a WordPress para registrar el camino.

<https://www.ks7000.net.ve>

[Smus](#) el 22 de agosto de 2012.

- Altamente recomendado (desempolvad los libros sobre Funciones Vectoriales): ["Fundamentals of image procesing"](#) por Jakub Szymanowski, Polonia.
-

Si tenemos en cuenta que las imágenes en una página web en promedio ocupan entre 60 y 70% del tamaño en bytes de la misma ***debemos prestar especial atención en escoger los formatos de imágenes correctos para ahorrar al usuario tiempo, dinero en conexión, y se queden más de un minuto en nuestro sitio web.***

Fuentes consultadas.

En idioma inglés.

- ["Serving WebP images for PNG and JPG files"](#)
- ["Writing Your First Code"](#) by Peter Leow.