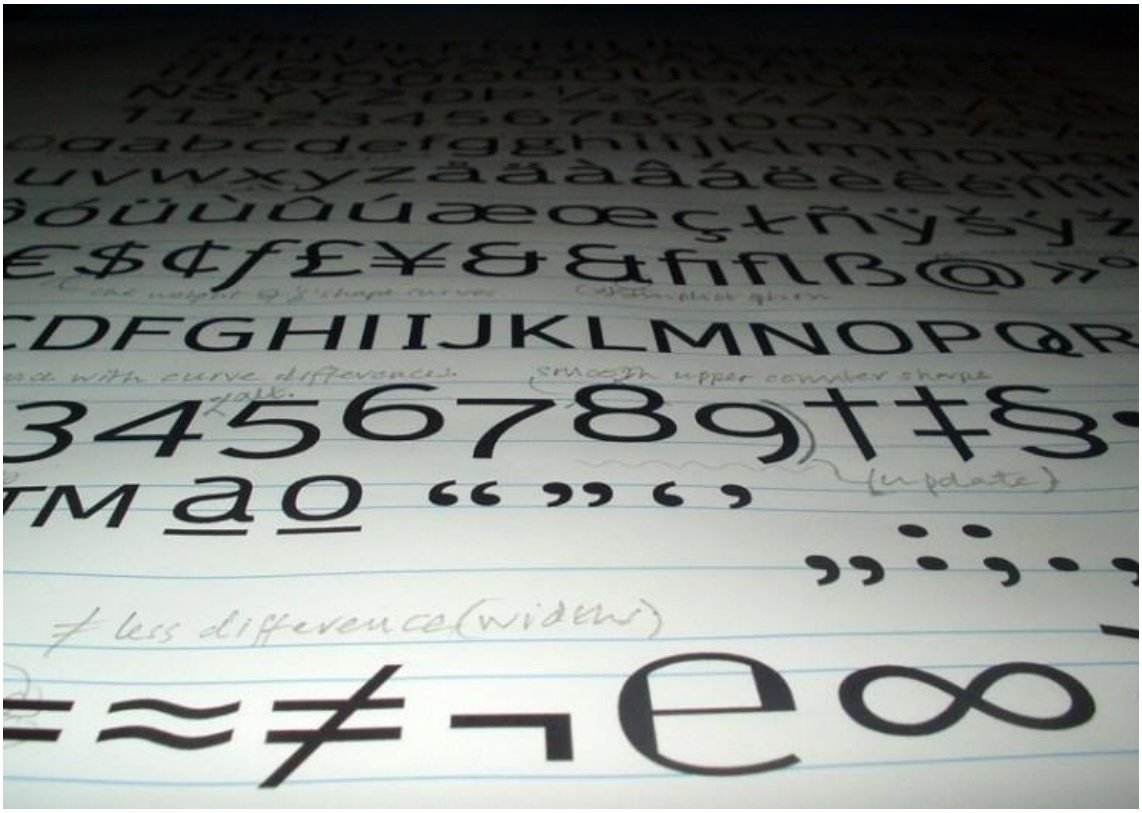
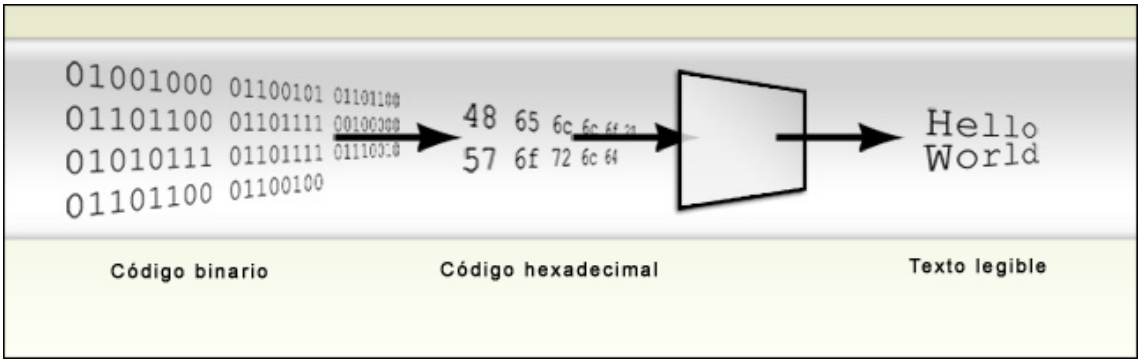
CHARSET: CODIFICACIÓN DE CARACTERES PARA GENERAR EMAILINGS EFICIENTES



Los equipos informáticos no hablan español, inglés ni chino, sino en código binario; para codificar todo lo que vemos en nuestras pantallas únicamente utilizan ceros y unos.

Cuando un programa abre un archivo lee todos los bytes y los transforma convirtiéndolos en una interpretación legible para los humanos: letras, números, símbolos y demás caracteres especiales.

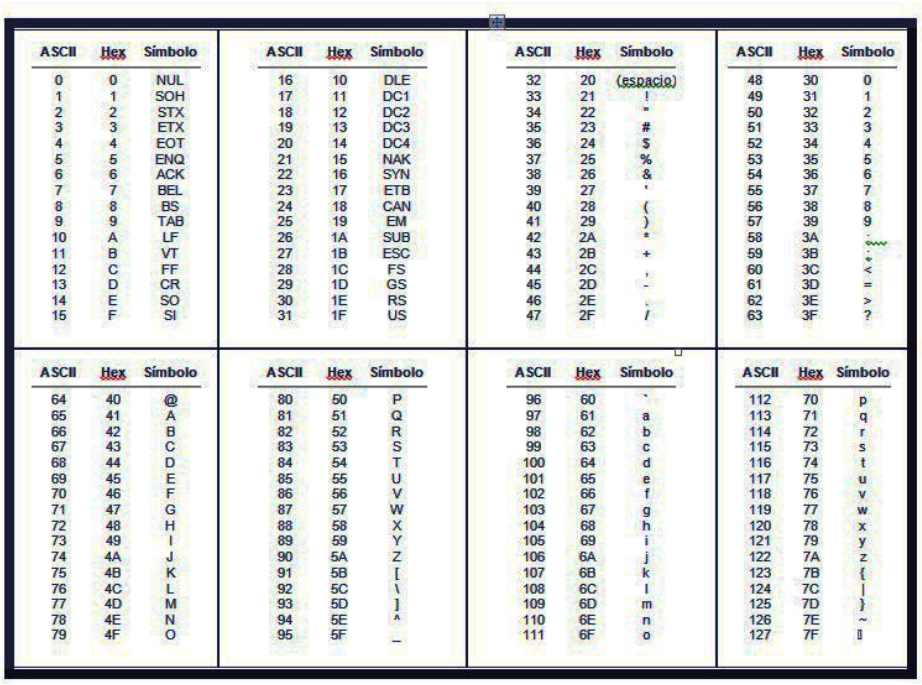


¿Pero cómo sabe la máquina que la combinación de ocho bits 01001101 por ejemplo debe mostrar una "M" o que cuando tecleamos la letra "p" debe transformarla en el byte 01110000? La respuesta la tenemos en la codificación de caracteres que estemos utilizando, o también conocido con el nombre de CharSet, del cual vamos a hablar en este libro blanco para que tus emailings se vean correctamente.

Primeros pasos: ASCII

Este juego de caracteres fue el primero que vio la luz, en una época en la que el almacenamiento era algo costoso y escaso, así que se apostó por que cada carácter sólo ocupara un byte.

Como mucho de vosotros sabréis, un byte está formado por 8 bits y cada uno de esos bits puede adoptar el valor de 0 o 1. De esta forma, 1 byte puede tener 256 valores diferentes (2^8), pero a pesar de esto, los creadores de ASCII decidieron utilizar sólo 7 para representar todos los caracteres que creían que eran necesarios para cualquier persona. En este grupo de caracteres se encontraban el alfabeto inglés, ciertos signos de puntuación (punto, coma, interrogación…) y otros elementos no imprimibles como el salto de línea o el tabulador.



ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
0	0	NUL	16	10	DLE	32	20	(espacio)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	▯

Tabla caracteres ASCII

Con el bit que no se utilizó para representar caracteres, se decidió utilizarlo como bit de paridad, un mecanismo para asegurar que la información era correcta.

Todo esto funcionó muy bien hasta que el resto del mundo quiso representar los caracteres que utilizaban en su lengua, ya que ASCII impedía la representación de letras como nuestra “ñ” o el alfabeto cirílico.

ISO y el uso del octavo bit

Ante el problema de no poder representar ciertos caracteres con el sistema ASCII, muchos vieron en el octavo bit, que hasta ahora era utilizado como bit de paridad, una forma de poder representar 128 nuevos códigos.

El desarrollo de este estándar trajo consigo la aparición de varios tipos, cada uno de los cuales era capaz de representar los caracteres de ciertos alfabetos. Entre los distintos tipos de codificación que nos podemos encontrar están los siguientes:

ISO-8859-1
También conocido con el sobrenombre de Latin-1. Se trata de una codificación muy utilizada en gran parte de Europa y el continente americano. Esta codificación incluye los caracteres desde la a-z (tanto minúsculas como mayúsculas), los números y símbolos de uso habitual, a excepción del símbolo del €.

ISO-8859-15
Se trata de un sistema de codificación similar al visto anteriormente, aunque incluye algunas diferencias como es la inclusión del símbolo del €.

ISO-8859-5
Representa al alfabeto cirílico con las que se pueden escribir en ruso, ucraniano o serbio.

ISO-8859-6
Es el estándar del alfabeto árabe. Comprende las letras básicas de la lengua árabe, aunque no incluye las extensiones necesarias para el persa ni el paquistaní. Aunque contiene las bases de del árabe, hay que tener en cuenta que las letras de esta lengua pueden tener hasta cuatro formas de representación diferente, por lo que para su correcta presentación en una página hace falta a menudo un programa independiente que analice el contexto en el que se encuentran las letras y le de la interpretación adecuada.

ISO-8859-7
Estándar que cubre todas las letras de la lengua griega.
El problema del ISO es precisamente su alto número de grupos, que hace que en muchas ocasiones no sepamos cuál es el adecuado para representar una lengua.

UNICODE, un sistema para aglutinar todas las codificaciones en una

Con el paso del tiempo y el aumento de uso de Internet que ha traído la conversión de documentos a un mayor número de lenguas, el estándar ISO se ha mostrado con el tiempo insuficiente para atender todos los caracteres utilizados en cualquier parte del mundo.

Así pues era necesaria una forma global de representar todos los posibles caracteres existentes y UNICODE fue el encargado de acometer esta tarea.

Este estándar fue desarrollado por la UTC y tiene como particularidad que no es un juego de caracteres, sino que es un estándar que se encarga de asignar un código numérico único a cada elemento que queremos representar.

A la hora de hacer referencia a un carácter Unicode, se hace utilizando el siguiente formato: U+XXXX, donde las X hacen referencia al código numérico del carácter en base hexadecimal. Por poner un ejemplo, la “Á” tiene el asignado el valor U+00C1. Podéis ver un listado de estos caracteres en el siguiente enlace.

Como hemos comentado anteriormente, UNICODE sólo es el encargado de indicar el valor que se le asigna a cada elemento que queremos representar pero no indica cómo se tiene que representar esos elementos de manera binaria, el idioma utilizado en los dispositivos informáticos. Es aquí donde aparece la codificación más utilizada en la actualidad: UTF-8.

Codificación UTF-8

UTF-8 es el formato de codificación de caracteres más utilizado en la actualidad y se caracteriza por utilizar un número de bytes variable dependiendo del carácter que se quiera representar. Pero además de esta característica, hay otras que lo hacen muy atractivo a la hora de ser utilizado en la codificación de páginas web, entre las que podemos destacar:

- Capaz de representar cualquier carácter Unicode.
- Utilización de 1 a 4 bytes para representar los distintos caracteres, dependiendo de su valor Unicode.
- Permite la representación de cualquier mensaje ASCII sin necesidad de tener que hacer cambio alguno en su codificación.

UTF-8 divide los caracteres Unicode en varios grupos, dependiendo del número de bytes necesarios para su codificación. Estos grupos son los siguientes:

- Caracteres codificados con un byte. Aquí están los que son representados en el formato ASCII.
- Caracteres codificados con dos bytes. Este grupo incluye los caracteres romances más signos diacríticos, y los alfabetos griego, cirílico, armenio, hebreo y árabe entre otros.
- Caracteres codificados con tres bytes. Formado por los caracteres del plano básico multilingüe de Unicode, que unido al grupo anterior, incluye la mayoría de caracteres de uso común en todo el mundo, incluido el chino, japonés y coreano.
- Caracteres codificados cuatro bytes. Aquí se encuentran los símbolos matemáticos y alfabetos clásicos para uso académico como el alfabeto persa, fenicio…

Configuración del parámetro Charset de HTTP

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
</head>
```

Hasta aquí, todo lo que hemos visto ha sido teoría, ¿pero cómo llevar esto a la práctica? ¿Cómo indicar el tipo de codificación que queremos utilizar a la hora de desarrollar una página web? La respuesta la tenemos en la etiqueta meta charset que nos ofrece HTML.

Esta etiqueta es de gran importancia ya que es la encargada de indicar al navegador el tipo de codificación que debe utilizar para representar la información de forma correcta y que todo se vea bien. Aunque es una etiqueta que en muchas ocasiones se ignora y no se pone, tal es su importancia que siempre debería de aparecer a la hora de diseñar cualquier aplicación web.

Por lo general, la línea del encabezado de HTTP que se utiliza para indicar el tipo de codificación a utilizar es la siguiente:

Content-Type: text/html; charset=utf-8

Pero además de indicarlo desde el código HTML de la web, también se puede indicar el tipo charset que queremos utilizar mediante la configuración de los servidores web, o bien mediante lenguajes de programación del lado del servidor.

1.- Configuración del servidor

Dependiendo del tipo de servidor que utilicemos, la configuración de este parámetro se lleva a cabo de una forma u otra.

a) Servidor Apache

En este caso se puede hacer por medio de la directiva AddCharset se puede indicar el tipo de charset a utilizar para un determinado sitio. Si lo que queremos es configurar el servidor para que todos los sitios hospedados utilicen un determinado charset, se puede hacer por medio de la directiva AddDefaultCharset.

b) Servidor IIS

En el servidor web desarrollado por Microsoft, para indicar el charset a utilizar, en el Administrador de servicios de Internet pulsaremos con el botón derecho en “Default Web Site”. Luego iremos a “Properties” => “Http Headers” => “File Types” => “New Type”. Ahí ingresaremos la extensión que deseamos mapear, normalmente serán las extensiones .html y .htm. Luego para Content type agregaremos “text/html; charset=utf-8”, pudiendo cambiar el valor utf-8 por otro que queramos utilizar.

2. Configuración mediante programación del lado del servidor

Como ya hemos comentado, el tipo de charset que queremos que se utilice para una determinada página, también puede ser indicado mediante el lenguaje de programación que utilicemos para la creación del portal. Veamos a continuación la forma de realizarlo dependiendo del lenguaje utilizado.

a) Perl

En Perl, para indicar el tipo de codificación se utiliza la función print antes de imprimir cualquier otra información. Es lo primero que debe aparecer.

```
print "Content-Type: text/html; charset=utf-8\n\n";
```

b) Python

La forma de indicarlo en este lenguaje es similar a la vista anteriormente, pero con la diferencia que no hace falta poner el “;” al final de la instrucción.

```
print "Content-Type: text/html; charset=utf-8\n\n";
```

c) PHP

En PHP, la función que nos permite realizar el cambio de codificación de caracteres es la función header(). Un ejemplo de uso sería el siguiente.

```
header ('Content-type: text/html; charset=utf-8');
```

d) Java Servlets

En el caso de estar desarrollando la web con Servlets de Java, lo que habría que utilizar sería la función setContentType del objeto response, método que deberíamos utilizar antes de obtener cualquier objeto Stream o Writer. Un ejemplo de uso sería el siguiente.

```
resource.setContentType ("text/html;charset=utf-8");
```

e) JSP

En el caso de estar utilizar JSP, el charset se puede indicar por medio del objeto "page", lo que hará que lo que se imprima por pantalla se convierta automáticamente a la codificación indicada.

```
<%@ page contentType="text/html; charset=UTF-8" %>
```

f) ASP

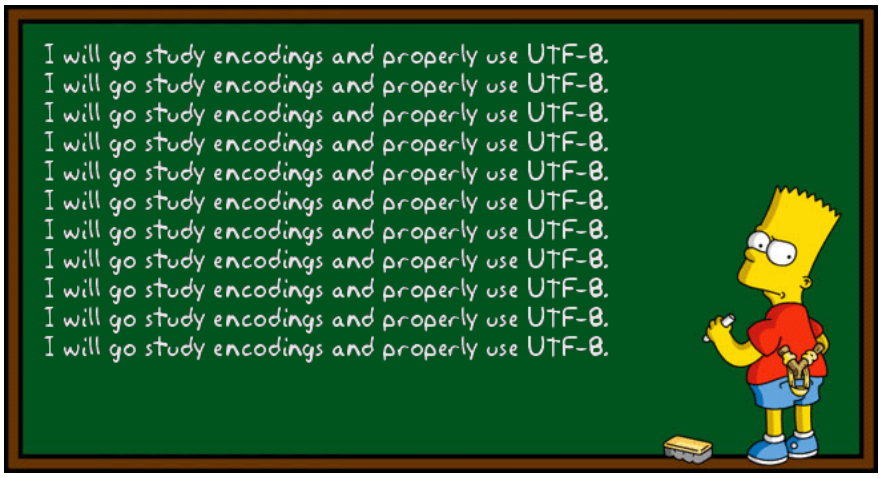
En el caso de ASP o ASP.NET, para indicar el charset que queremos utilizar, tenemos que utilizar la directiva charset del objeto Response, tal y como podéis ver a continuación.

```
<%Response.charset="utf-8"%>
```

Por último nos gustaría hablar de aquellos casos en los que aunque hayamos indicado bien el tipo de codificación a utilizar, aquellas palabras que lleven acentos o eñes, se vean con símbolos raros. Esto puede ser debido al tipo de codificación que se haya utilizado para la creación del archivo.

A continuación os dejamos un par de estas situaciones que pueden producir más de un dolor de cabeza a la hora dar con la solución, cuando con un simple cambio de formato del archivo sería suficiente para solucionarlo.

- Si se ve ◆ o □ => Archivo codificado ISO-8859-1, visto como UTF-8.
- Si se ven dos caracteres raros como Ãi => Archivo codificado UTF-8, visto como ISO-8859-1



Si vamos a empezar la creación de un sitio web, nuestra recomendación es que se apostara por hacer uso del sistema UTF-8, ya que nos ofrece la posibilidad de poder representar prácticamente cualquier tipo de carácter que nos podamos encontrar en todo el mundo