作者：邓乃文

字符编码

1. 字符编码背景.....2
2. 字符编码概念.....2
    a) 什么是字符编码.....2
    b) ASCII 码.....2
    c) ANSI.....3
    d) Unicode.....3
3. 常用的字符编码集及其区别.....4
    a) gb2312.....4
    b) gbk.....4
    c) utf-8.....4
    d) ISO-8859-1.....4

1. 字符编码背景

a) 为什么要学习字符编码

我们在学习中，经常会需要在文件中显示文字，也许是中文，英文，或者韩文等其他语言的文字。而有的同学可能遇到在编写文件(比如说 HTML)时，不能够正确的显示所写的文字，而显示出一堆看不懂的文字，如下图所示：

ä½ å¥½ï¼Œè¿™æ˜¯ä¸€æ®µä¸æ–‡

有同学可能知道在 HTML 中我们可以利用 meta 标签来解决问题，一般是用了 utf-8 字符编码。那什么是字符编码？

b) 用计算机表示字符

我们都知道计算机起源于美国，早期的计算机只是用于科学计算。但是在计算机迅速发展时，计算机被要求不仅仅能够进行数值计算，还要进行字符处理和表示。所以需要让计算机能够识别文字并能够处理和显示。但计算机只能够识别 0 和 1，美国人则发明了一种表示字符的方式，即利用多个 0 和 1 的不同组合来表示不同的字符，比如字母 a、数字 1 等等。

c) 字符编码需要统一

早期美国因为计算机有很多台，不可能每台计算机用不同的 0 和 1 组合表示字符，所以不同计算机处理字符的方式得到统一的问题必须解决。而字符编码则应运而生。在 utf-8 中 0101 表示 b，在 gbk 中 0101 可能表示数字 2

2. 字符编码概念

a) 什么是字符编码

我们可以简单的理解为一套用于在计算机中用 0 和 1 的不同组合来表示字符的规则。即规定在计算机中 0 和 1 的哪种组合方式表示现实中的哪个字符。常见的字符编码有 ASCII、gbk、utf-8 等。

b) ASCII 码

早期美国为了解决字符编码问题，一套名为 ASCII（American Standard Code for Information Interchange，美国信息互换标准代码）码的编码方式被创造而出，使用 7 位(bit)来表示一个字符，总共能够表示 128 种字符（bit 是计算机中的最小数据单位，一个 bit 能够保存一个 0 或者 1）。

这套编码包含了控制码(例如\n 换行符等)、符号(?!等)、数字和 26 个英文字母包括其大小写。但是由于这是一套由美国人提出的美国字符编码所以并没有包含中文、韩文等等其他国家的语言，毕竟当时谁也没想到计算机会普及到全世界。

以下图片是 ASCII 码能够表示的字符集合：

ASCII表

（ American Standard Code for Information Interchange 美国标准信息交换代码 ）

高四位 / 低四位		ASCII控制字符												ASCII打印字符															
		0000						0001						0010		0011		0100		0101		0110		0111					
		0						1						2		3		4		5		6		7					
		十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	Ctrl	代码	转义字符	字符解释	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	Ctrl
0000	0	0		^@	NUL	\0	空字符	16	▶	^P	DLE		数据链路转义	32		48	0	64	@	80	P	96	`	112	p				
0001	1	1	☺	^A	SOH		标题开始	17	◀	^Q	DC1		设备控制 1	33	!	49	1	65	A	81	Q	97	a	113	q				
0010	2	2	☻	^B	STX		正文开始	18	↕	^R	DC2		设备控制 2	34	"	50	2	66	B	82	R	98	b	114	r				
0011	3	3	♥	^C	ETX		正文结束	19	‼	^S	DC3		设备控制 3	35	#	51	3	67	C	83	S	99	c	115	s				
0100	4	4	♦	^D	EOT		传输结束	20	¶	^T	DC4		设备控制 4	36	$	52	4	68	D	84	T	100	d	116	t				
0101	5	5	♣	^E	ENQ		查询	21	§	^U	NAK		否定应答	37	%	53	5	69	E	85	U	101	e	117	u				
0110	6	6	♠	^F	ACK		肯定应答	22	▬	^V	SYN		同步空闲	38	&	54	6	70	F	86	V	102	f	118	v				
0111	7	7	•	^G	BEL	\a	响铃	23	↨	^W	ETB		传输块结束	39	'	55	7	71	G	87	W	103	g	119	w				
1000	8	8	◘	^H	BS	\b	退格	24	↑	^X	CAN		取消	40	(	56	8	72	H	88	X	104	h	120	x				
1001	9	9	○	^I	HT	\t	横向制表	25	↓	^Y	EM		介质结束	41	)	57	9	73	I	89	Y	105	i	121	y				
1010	A	10	◙	^J	LF	\n	换行	26	→	^Z	SUB		替代	42	*	58	:	74	J	90	Z	106	j	122	z				
1011	B	11	♂	^K	VT	\v	纵向制表	27	←	^[	ESC	\e	溢出	43	+	59	;	75	K	91	[	107	k	123	{				
1100	C	12	♀	^L	FF	\f	换页	28	∟	^\	FS		文件分隔符	44	,	60	<	76	L	92	\	108	l	124	|				
1101	D	13	♪	^M	CR	\r	回车	29	↔	^]	GS		组分隔符	45	-	61	=	77	M	93	]	109	m	125	}				
1110	E	14	♫	^N	SO		移出	30	▲	^^	RS		记录分隔符	46	.	62	>	78	N	94	^	110	n	126	~				
1111	F	15	☼	^O	SI		移入	31	▼	^-	US		单元分隔符	47	/	63	?	79	O	95	_	111	o	127	⌂	^Backspace 代码：DEL			

注：表中的ASCII字符可以用“Alt + 小键盘上的数字键”方法输入。 2013/08/08

其中 0~127 是十进制数据。如当用 ASCII 码来表示十进制的 0 时，实际会显示出空格，十进制 33 会显示出！

(而计算机中是用二进制来表示十进制的，所以对于字符编码来说，十进制就是起的一个中间的作用，最终还是用的二进制来表示字符的。对于二进制和十进制的相关内容，我们会在 JavaScript 中讲到或者可以自行百度进行学习)

但是 ASCII 码是美国最早开发的字符编码集，并没有考虑到其他国家的语言，所以 ASCII 码并不能解决我们的问题。

c) ANSI

当计算机普及到全世界时，各个国家面临的首要问题就是要针对自己国家的语言制定一套自己国家的编码规范，我国就提出里一套针对中文的 GB2312 的编码方式，这套编码方式基于 ASCII 码，其使用 2 个字节表示一个汉字，具体的方式是前 127 个字符和 ASCII 码保持不变。当第一个字节(高字节)大于 160 的时候，表示一个汉字的开始，再用这个字节组合第二个字节(低字节，范围也是 160-255)共同表示一个汉字。在这套编码方式中，不仅把中文编码进去，还把一些数学符号、罗马希腊字母和日本假名等等都编码进去，并且还把 ASCII 中原有的 26 个英文字母和符号都编入，当然这些字母是以 2 个字节表示，为和 ASCII 中原有的字母区别表示，称前者为“全角字符”，后者为“半角字符”。

当然我大中华文化底蕴深厚，GB2312 也只能编入部分常用汉字，为了把更多的汉字编入进来，针对 GB2312 进行扩充，就创造出了 GBK 标准。GBK 只要求当高字节大于 127 时就表示汉字的开始，低字节也不再要求范围。

类似我国的编码方案，其他地区和国家也制定了自己的编码方案，如日本的 Shift_JIS 等等。这些编码方案称为 "DBCS"（Double Byte Charecter Set 双字节字符集）"即是用双字节表示一个字符，也称为 ANSI。这里的 ANSI 代表了不同国家的不同编码方案，如果一台 Windows 操作系统设定为中文，那么 ANSI 就表示 GBK，如果设定为日本 ANSI 就表示 Shift_JIS。ANSI 虽然能够表示全世界的字符，但是产生一个麻烦的问题：

当使用 ANSI 编码的一篇文章，被台湾友人或者外国友人拿去，打开后，全是乱码，因为台湾地区使用的是 BIG5 码，不同的地区编码虽然都称作 ANSI，但是互相之间没有算法做出转换，这大大影响了阻碍各地区、国家之间的交流。

d) Unicode

为了统一全世界的文字编码，ISO（国际标准化组织）制定了一种新的编码规范，这种编码规范将全世界的文字放在一张表内，称它为"Universal Multiple-Octet Coded Character Set"，简称 UCS, 俗称"UNICODE"。请注意，Unicode 和 GBK、BIG5 等 ANSI 编码仍然没有一种直接的算法进行转换，Unicode 在制定时可以看成是废了所有的地区性编码，重新制定的编码，也不是从 ASCII 继承而来(但是前 128 字符仍保留为 ASCII 字符)。

严格意义上 Unicode 只是一套标准，为全世界文字给予一个唯一的编码，但是并没有规定在计算机中如何存储。而我们所认识和熟悉的 utf-8、utf-16、utf-32 则是 Unicode 规则的实现，也就是基于 Unicode 而制定的一套可用的字符编码。同时为了和 ASCII 码兼容，其前 127 个位置表示和 ASCII 码一致。也就是说，如果替换成了 Unicode 码，那么原来使用 ASCII 码的部分内容也不会受影响。

3. 常用的字符编码集及其区别

a) gb2312

基于 ASCII 码进行开发的中文字符编码规则。而 ASCII 码是以 1 个字节来表示字符，一个字节最多能有 255 种 0 和 1 的表示组合，即最多能够表示 255 个字符，而这个数量还远远不够，所以 gb2312 就规定使用 2 个字节来表示文字。这样，所有常用的汉字就可以包含在里面了。但只能够识别中文和 ASCII 码中的字符，其他语言文字不识别。

b) gbk

GBK 是国家标准 GB2312 基础上扩容后兼容 GB2312 的标准。GBK 的文字编码是用双字节来表示的，即不论中、英文字符均使用双字节来表示，为了区分中文，将其最高位都设定成 1。GBK 包含全部中文字符，是国家编码，通用性比 UTF8 差，不过 UTF8 占用的数据库比 GBK 大。但一样只能识别英文和亚洲语言，不能很好识别西方国家的语言。

c) utf-8

UTF-8（8 位元，Universal Character Set/Unicode Transformation Format）是基于 Unicode 的一种可变长度字符编码。它可以用来表示 Unicode 标准中的任何字符，而且其编码中的第一个字节仍与 ASCII 相容，使得原来处理 ASCII 字符的软件无须或只进行少部份修改后，便可继续使用。

utf-8 使用 1~4 个字节来表示字符，而具体用几个字节来表示则是按照变长编码规则来确定。utf-8 的变长编码格式会保证一直以最优的式来存储数据。比如 UTF8 用一个字节存储英文字母，一般中文用两个字节来表示。之后 的字符用 3~4 个字节来表示。所以 UTF8 对于存储英文字母的高效率来源于对之后字符保存效率的牺牲。这里的合理性在于：如果待保存的文本中字符大多数为英文字母，则存储效率能够提高，因为大多数字符都是采用一个字节保存。

而 utf-8 和 utf-16 以及 utf-32 的主要区别有以下几点：

1. utf-8 用 1~4 个字节保存数据。utf-32 固定使用 4 个字节来表示字符。utf-16 使用 2 或 4 个字节进行存储。

2. 整体而言 utf-8 具有一定灵活性，也是大多数计算机首选。

d) ISO-8859-1

ISO-8859-1 编码是单字节编码，向下兼容 ASCII，其编码范围是 0x00-0xFF，0x00-0x7F 之间完全和 ASCII 一致，0x80-0x9F 之间是控制字符，0xA0-0xFF 之间是文字符号。支持西欧国家的语言。、

ISO-8859-1 使用一个字节来保存数据。但不支持亚洲国家语言，但实际很多编程语言使用了 iso-8859-1 作为编码规则。

e) 对于几种常见字符编码的总结

1. ASCII 码：占一个字节，只能包含 128 个符号，不能表示汉字

2. ISO-8859-1：（latin-1）：占一个字节，收录西欧语言，不能表示汉字，

3. ANSI：占两个字节，在简体中文的操作系统中 ANSI 就指的是 GB2312.

4. GBK:对 gb2312 的补充，支持更多的汉字，但不支持西方语言

5. utf-8：用 1~4 个字节存储数据，基于 Unicode，能够以较优的性能表示世界上所有的文字。