JavaScript 常见编码知识

1 ASCII 字符集

2 非 ASCII 字符集

  2.1 GB2312 字符集

  2.2 GBK 字符集

  2.3 GB18030 字符集

3 Unicode

  3.1 UTF-8

  3.2 UTF-16

  3.3 UTF-32

  3.4 字节顺序标记

4 Base64 编码

5 编码转换 & 编码方法

  5.1 编码转换

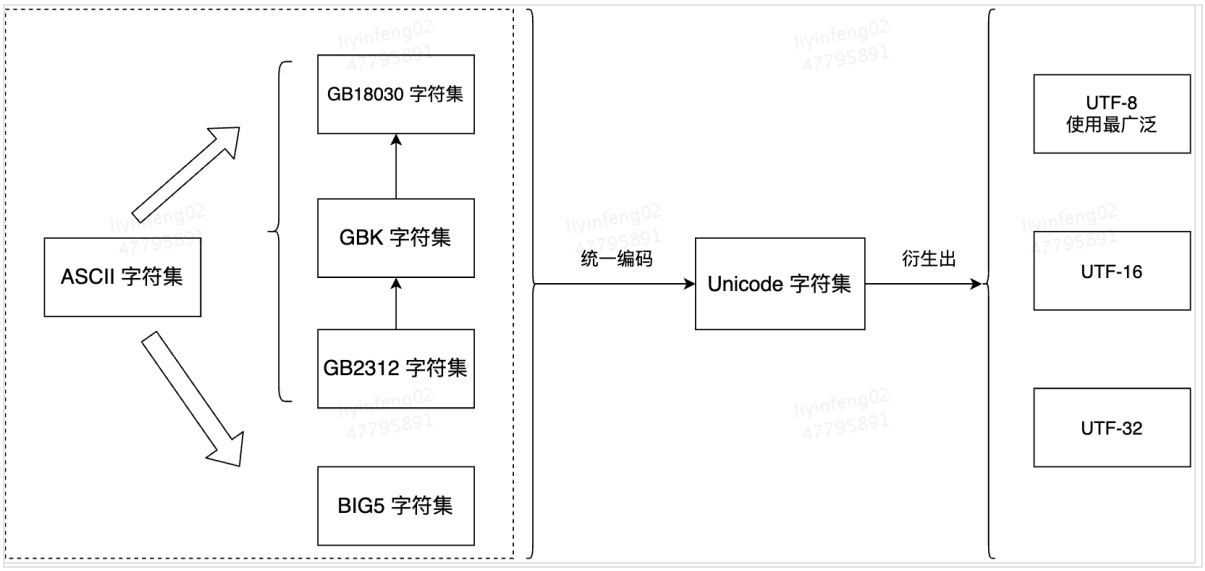
  5.2 编码方法

6 总结

7 参考

最近在看文件操作相关的知识，里面涉及到字符编码相关的知识，想接触计算机以来，接触过各种编码：ASCII、GB2312、UTF-8、Base64 等等。为什么有这么多种编码方式？编码发展历程又是怎么样呢？下面就一起去学习一下。

下面是自己整理的一个编码相关的流程示意图：



1 ASCII 字符集

ASCII：美国信息交换标准代码。20 世纪 60 年代，美国人发明的。

- 收录 128 个字符，0~31 和 127 是非打印字符，32~126 是 打印字符。
- 为什么是 128 个字符，使用二进制标识，是使用 1 个 Byte（8 个 Bite）进行标识，2^8 = 128。

> 查看完整 ASCII 码对照表，可以查看：脚本之家 – ASCII 码对照表

2 非 ASCII 字符集

上面说 ASCII 编码一共是 128 个， 英语的表示是够了，但是表示其他语言的话，128 个字符远远是不够的。不同国家就同一个字母的表示方式是不同的，语言特有的字符更是数不胜数，就汉字来说，多达 10 万个，常用汉字多达六千多个。一个字节表示肯定是远远不够的，因此使用其他方式进行解决。

2.1 GB2312 字符集

GB2312：中国国家标准简体中文字符集。中国计算机科学家设计了一种字符集，收录了汉字、拉丁文字母、希腊字母等多种文字及字母。公共包含汉字 6763 个，其他文字符号 682 个（注意：GB2312 字

符集并没有收录 ASCII 字符集中的字符）。

具体编码方式与上面编码有所区别，因为涉及的字符太多了。使用的是 分区概念。一共划分 94 个区，每个区收录 94 个字符。所以定位方式就是放在第几区的第几个字符。汉字“啊”放在 16 区的第一位，因此对应的数字是 1601（即区位码是 1601）。

- 01~09区（682个）：特殊符号、数字、英文字符、制表符等，包括拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母等在内的682个全角字符；
- 10~15区：空区，留待扩展；在附录3，第10区推荐作为 GB 1988−80 中的94个图形字符区域（即第3区字符之半形版本）。
- 16~55区（3755个）：常用汉字（也称一级汉字），按拼音排序；
- 56~87区（3008个）：非常用汉字（也称二级汉字），按部首/笔画排序；
- 88~94区：空区，留待扩展。

> 1. 查看 某个汉字的区位码，可以查看：GB2312区位码查询与转换
> 2. 查看全部 GB2312 字符集，可以参考： GB2312简体中文编码表

2.2 GBK 字符集

GBK： 汉字内码扩展规范。中国汉字太多，许多不太经常使用的字符没有收录，台湾香港使用的繁体字也没有进行收录，GBK 字符集就是在 GB2312 字符集基础上，对收录字符进行了补充。

- GBK 向下完全兼容 GB2312−80编码（包含 GB 2312 中的全部汉字、非汉字符号）
- BIG5 字符集中的全部汉字（普及于台湾、香港、澳门等繁体中文区域，但并非标准）
- 与 ISO 10646 相应的国家标准 GB 13000 中的其它 CJK 汉字，以上合计 20902 个汉字
- 其它汉字、部首、符号，共计 984 个

2.3 GB18030 字符集

GB 18030：国家标准 GB 18030−2005《信息技术中文编码字符集》。是中华人民共和国现时最新的内码字集。

- GB 18030 与 GB 2312−1980 和 GBK 兼容，共收录汉字70244个
- 采用多字节编码，每个字可以由 1 个、2 个或 4 个字节组成

- 编码空间庞大，最多可定义 161 万个字符
- 支持中国国内少数民族的文字，不需要动用造字区
- 汉字收录范围包含繁体汉字以及日韩汉字

3 Unicode

Unicode：联盟官方中文名称为统一码，台湾官方中文名称为万国码，也译为国际码、单一码，是计算机科学领域的业界标准。

就上面我们了解了这么多的编码方式，同一个二进制的数字能被解析成不同的符号。因为当打开一个文件必须准确的知道编码方式，否则就可能出现因编码方式错误，导致的乱码现象。

那么就需要有一种编码，能将世界上全部符号进行收纳，统一给每个符号一个独一无二的编码。这样才能彻底解决乱码问题。因此 Unicode【国际码】 成为了国际认可的编码标准。

Unicode 是个很大的集合，现在可容纳 100 多完符号，每个符号的编码都不同。汉字”花“编码为：U+82B1。

> 1. 在线 Unicode 查询，可参考：Unicode 查询
> 2. 中日韩 Unicode 编码表，可查询：中日韩文字 Unicode 编码表

虽然 Unicode 是编码标准，存在问题：

1. 如何区分 Unicode 与 ASCII ？
   ❓ &：ASCII 编码：00100110；Unicode 编码：U+0026
2. 英文使用一个字节就能表示了，但是按照 Unicode 统一规定，每个符号要用三个或4个字节才能表示。对于存储来说太过浪费了，是无法接受的。

造成的后果：

1. 出现了多种存储方式，有许多不同二进制格式表示 Unicode；
2. Unicode 很长时间不能推广。

3.1 UTF-8

UTF-8：一种针对 Unicode 的可变长度字符编码，也是一种前缀码。互联网的普及促使出现一种统一编码，UTF-8 就是互联网上使用最广的一种 Unicode 的实现方式，其他的实现方式还包含了：UTF-

16（使用 2~4 个字节表示）、UTF-32（使用4个字节表示），不过互联网上基本不使用。

特点：一种变长的编码方式（128 个 ASCII 字符集使用一个字节，其他随符号不同而变化）

编码规则

- 先把编号转成二进制，再按照下面规则填入，位数不够补 0
- 单字节符号，第一位设为 0；后边7位是这个符号的 Unicode 码，因此英语字母 UTF-8编码与ASCII码相同
- 大于单字节n 位，第一个字节前n位都是1，n+1位是0，后边字节前两位都是10
- 其余没有提及的为禁止，全部是这个符号的 Unicode 码。

Unicode Segment	Encoding
`0x000000` - `0x00007f`	`0xxxxxxx`
`0x000080` - `0x0007ff`	`110xxxxx` `10xxxxxx`
`0x000800` - `0x00ffff`	`1110xxxx` `10xxxxxx` `10xxxxxx`
`0x010000` - `0x10ffff`	`11110xxx` `10xxxxxx` `10xxxxxx` `10xxxxxx`

举例

- 以大写字母 P 为例，Unicode 编号：U+0050，二进制是：101 0000，属于单字节符号。首位设为 0 即可。
  因此，UTF-8 编码为：0101 0000（即：50）。
- 以汉字 严 为例，Unicode 编号：U+4E25，二进制是：100 1110 0010 0101，占两个字节，属于上面第三行。那么编码需要 3 个字节，UTF-8 编码为：11100100 10111000 10100101（即：E4B8A5）。

3.2 UTF-16

Code Point Segment	Encoding
`0x000000` - `0x00ffff`	`xxxxxxxx xxxxxxxx`
`0x010000` - `0x10ffff`	`110110yy yyyyyyyy 110111xx xxxxxxxx`

概念：编码规则实在没看懂，后边在进行补充吧！

优点：大部分字符使用2个字节就能实现表示。

缺点：不能兼容 ASCII 编码

3.3 UTF-32

Code Point Segment	Encoding
`0x000000` - `0x10ffff`	`xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx`

概念：使用固定长度的字节表示字符（4 个字节 – 32 个 Byte）。

缺点：空间浪费极大，基本不会使用。

3.4 字节顺序标记

- Be 编码（大端序）：高字节在左边，低字节在右边
- Le 编码（小端序）：高字节在右边，低字节在左边

对应编码及文件前缀，如下：

UTF Encoding	Byte Order Mark
UTF-8	`EF BB BF`
UTF-16 LE	`FF FE`
UTF-16 BE	`FE FF`
UTF-32 LE	`FF FE 00 00`
UTF-32 BE	`00 00 FE FF`

举例

以严为例（Unicode 编号：U+4E25，二进制是：0100 1110 0010 0101）：

- 大端序编码：FE FF 4E 25
- 小端序编码：FF FE 25 4E
- UTF–8 编码（十六进制）：EF BB BF E4 B8 A5

4 Base64 编码

Base64 是一组相近的二进制编码规则。为什么要使用 Base64 编码呢？主要是计算机的数据基本都是按照 ASCII 进行存储的，为了保证数据传输的完整性与传输过程中数据不丢失，会先将字符转成 Base64 之后再进行传输。

特点：每个字符占 6 个字符，不能整除的话补充 0。快速预览表 如下👇：

Index	Binary	Char	Index	Binary	Char	Index	Binary	Char	Index	Binary	Char
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
10	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/
padding		=									

（引用自：https://mp.weixin.qq.com/s/QHi6BVM5Jt8XwZ_FKcRYsg）

5 编码转换 & 编码方法

5.1 编码转换

- Base64、UTF-8、ASCII 关系
  - utf-8 -> base64(编码) -> ASCII
  - ASCII -> base64(解码) -> utf-8

5.2 编码方法

- encodeURI()：将特定字符进行转义编码，转义字符：除（保留类型 + 非转义字符 + 数字符号）之外。
- decodeURI()：将👆 方法转义的字符进行解码，反转义字符同上。
- encodeURIComponent()：编码，转义字符：除非转义类型之外其他字符。
- decodeURIComponent()：解码，反转义字符同上。

> 注
>
> - 区别
>   - encodeURI 和 decodeURI 函数操作的是完整的 URI；这俩函数假定 URI 中的任何保留字符都有特殊意义，所有不会编码它们。
>   - encodeURIComponent 和 decodeURIComponent 函数操作的是组成 URI 的个别组件；这俩函数假定任何保留字符都代表普通文本，所以必须编码它们，所以它们出现在组成一个完整 URI 的组件里面时不会解释成保留字符了。
> - URL：**称为统一资源标识符**，是用来标识互联网上的资源和怎么访问这些资源的传输协议的字符串。
>
>   组成形式：*Scheme* : *First* / *Second* ; *Third* ? *Fourth*
> - 保留类型：; , / ? : @ & = + $
> - 非转义字符：字母 数字 - _ . ! ~ * ' ( )
> - 数字符号：#

- btoa()：返回 Base64 表示的字符串。btoa('P') // 'UA=='
- atob()：返回 ASCII 表示的编码。atob('UA==') // 'P'
  注意：以上两个方法对于非有效字符串，会抛出 DOMException

- charCodeAt()：返回的是 0~65535 之间的整数。0x[字符/汉字的 Unicode 编码]
  举例 1：【是】Unicode 编码：U+662F，'是'.charCodeAt() = 26159(0x662F)
  举例 2：【s】Unicode 编码：U+0073，'s'.charCodeAt() = 115 (0x73)

6 总结

以上就是涉及编码相关的全部知识，对于UTF-16、UTF-32 写的不是很详细，在我们实际运算中实际也是用的比较少的。大家有兴趣可以通过链接进行详细研究。

后续涉及到相关的知识，会在进行补充。希望大家读完本文有所收获，谢谢！

最后举一个例子

文本	P	严
Unicode 编码	U+0050 二进制：0101 0000	U+4E25 二进制：100 1110 0010 0101
UTF-8 编码	0101 0000	11100100 10111000 10100101
Base64 编码	UOS4pQ==	
ASCLL 编码	80	

> 在线编码解码工具：http://tool.haooyou.com/code

7 参考

> 阮一峰：字符编码笔记：ASCII，Unicode 和 UTF-8
> 字符集与编码
> 维基百科
> 常见的字符编码ascii、gb2312、utf-8和base64的规则