关于URL编码

作者： [阮一峰](http://www.ruanyifeng.com/)

日期： [2010年2月11日](http://www.ruanyifeng.com/blog/2010/02/)

**一、问题的由来**

URL就是网址，只要上网，就一定会用到。



一般来说，URL只能使用英文字母、阿拉伯数字和某些标点符号，不能使用其他文字和符号。比如，世界上有英文字母的网址"http://www.abc.com"，但是没有希腊字母的网址"http://www.aβγ.com"（读作阿尔法-贝塔-伽玛.com）。这是因为网络标准[RFC 1738](http://www.ietf.org/rfc/rfc1738.txt)做了硬性规定：

"...Only alphanumerics [0-9a-zA-Z], the special characters "$-\_.+!\*'()," [not including the quotes - ed], and reserved characters used for their reserved purposes may be used unencoded within a URL."

"只有字母和数字[0-9a-zA-Z]、一些特殊符号"$-\_.+!\*'(),"[不包括双引号]、以及某些保留字，才可以不经过编码直接用于URL。"

这意味着，如果URL中有汉字，就必须编码后使用。但是麻烦的是，RFC 1738没有规定具体的编码方法，而是交给应用程序（浏览器）自己决定。这导致"URL编码"成为了一个混乱的领域。

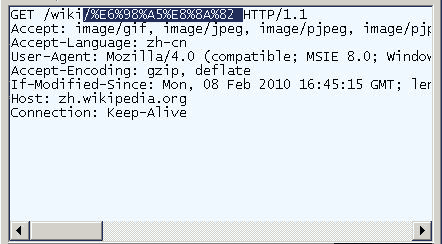
下面就让我们看看，"URL编码"到底有多混乱。我会依次分析四种不同的情况，在每一种情况中，浏览器的URL编码方法都不一样。把它们的差异解释清楚之后，我再说如何用Javascript找到一个统一的编码方法。

**二、情况1：网址路径中包含汉字**

打开IE（我用的是8.0版），输入网址"**http://zh.wikipedia.org/wiki/春节**"。注意，"春节"这两个字此时是网址路径的一部分。



查看HTTP请求的头信息，会发现IE实际查询的网址是"**http://zh.wikipedia.org/wiki/%E6%98%A5%E8%8A%82**"。也就是说，IE自动将"春节"编码成了"%E6%98%A5%E8%8A%82"。



我们知道，"春"和"节"的utf-8编码分别是"E6 98 A5"和"E8 8A 82"，因此，"%E6%98%A5%E8%8A%82"就是按照顺序，在每个字节前加上%而得到的。（具体的转码方法，请参考我写的[《字符编码笔记》](http://www.ruanyifeng.com/blog/2007/10/ascii_unicode_and_utf-8.html)。）

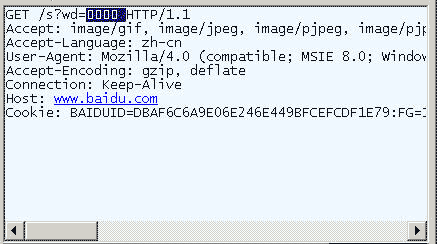
在Firefox中测试，也得到了同样的结果。所以，**结论1就是，网址路径的编码，用的是utf-8编码。**

**三、情况2：查询字符串包含汉字**

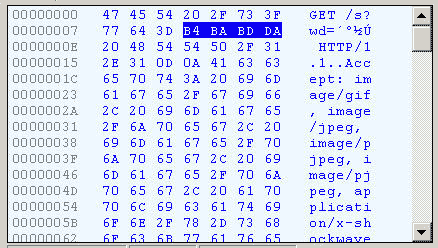
在IE中输入网址"**http://www.baidu.com/s?wd=春节**"。注意，"春节"这两个字此时属于查询字符串，不属于网址路径，不要与情况1混淆。



查看HTTP请求的头信息，会发现IE将"春节"转化成了一个乱码。

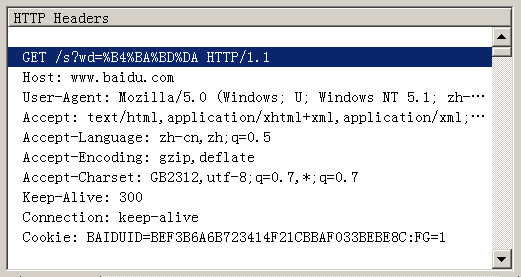


切换到十六进制方式，才能清楚地看到，"春节"被转成了"B4 BA BD DA"。



我们知道，"春"和"节"的GB2312编码（我的操作系统"Windows XP"中文版的默认编码）分别是"B4 BA"和"BD DA"。因此，IE实际上就是将查询字符串，以GB2312编码的格式发送出去。

Firefox的处理方法，略有不同。它发送的HTTP Head是"wd=%B4%BA%BD%DA"。也就是说，同样采用GB2312编码，但是在每个字节前加上了%。



所以，**结论2就是，查询字符串的编码，用的是操作系统的默认编码。**

**四、情况3：Get方法生成的URL包含汉字**

前面说的是直接输入网址的情况，但是更常见的情况是，在已打开的网页上，直接用Get或Post方法发出HTTP请求。

根据台湾中兴大学[吕瑞麟老师的试验](http://web.nchu.edu.tw/~jlu/classes/xml/ajax/urlencoding.shtml)，这时的编码方法由网页的编码决定，也就是由HTML源码中字符集的设定决定。

　　<meta http-equiv="Content-Type" content="text/html;charset=xxxx">

如果上面这一行最后的charset是UTF-8，则URL就以UTF-8编码；如果是GB2312，URL就以GB2312编码。

举例来说，百度是GB2312编码，Google是UTF-8编码。因此，从它们的搜索框中搜索同一个词"春节"，生成的查询字符串是不一样的。

百度生成的是%B4%BA%BD%DA，这是GB2312编码。



Google生成的是%E6%98%A5%E8%8A%82，这是UTF-8编码。



所以，**结论3就是，GET和POST方法的编码，用的是网页的编码。**

**五、情况4：Ajax调用的URL包含汉字**

前面三种情况都是由浏览器发出HTTP请求，最后一种情况则是由Javascript生成HTTP请求，也就是Ajax调用。还是根据吕瑞麟老师的文章，在这种情况下，IE和Firefox的处理方式完全不一样。

举例来说，有这样两行代码：

　　url = url + "?q=" +document.myform.elements[0].value; // 假定用户在表单中提交的值是"春节"这两个字

　　http\_request.open('GET', url, true);

那么，无论网页使用什么字符集，IE传送给服务器的总是"q=%B4%BA%BD%DA"，而Firefox传送给服务器的总是"q=%E6%98%A5%E8%8A%82"。也就是说，**在Ajax调用中，IE总是采用GB2312编码（操作系统的默认编码），而Firefox总是采用utf-8编码。这就是我们的结论4。**

**六、Javascript函数：escape()**

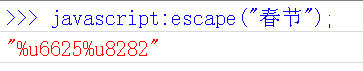
好了，到此为止，四种情况都说完了。

假定前面你都看懂了，那么此时你应该会感到很头痛。因为，实在太混乱了。不同的操作系统、不同的浏览器、不同的网页字符集，将导致完全不同的编码结果。如果程序员要把每一种结果都考虑进去，是不是太恐怖了？有没有办法，能够保证客户端只用一种编码方法向服务器发出请求？

回答是有的，就是使用Javascript先对URL编码，然后再向服务器提交，不要给浏览器插手的机会。因为Javascript的输出总是一致的，所以就保证了服务器得到的数据是格式统一的。

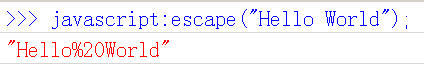
Javascript语言用于编码的函数，一共有三个，最古老的一个就是escape()。虽然这个函数现在已经不提倡使用了，但是由于历史原因，很多地方还在使用它，所以有必要先从它讲起。

实际上，escape()不能直接用于URL编码，它的真正作用是返回一个字符的Unicode编码值。比如"春节"的返回结果是%u6625%u8282，也就是说在Unicode字符集中，"春"是第6625个（十六进制）字符，"节"是第8282个（十六进制）字符。



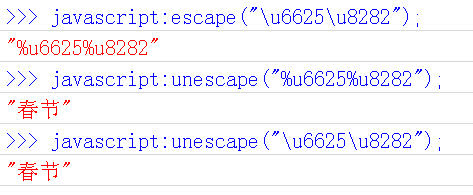
它的具体规则是，除了ASCII字母、数字、标点符号"@ \* \_ + - . /"以外，对其他所有字符进行编码。在\u0000到\u00ff之间的符号被转成%xx的形式，其余符号被转成%uxxxx的形式。对应的解码函数是unescape()。

所以，"Hello World"的escape()编码就是"Hello%20World"。因为空格的Unicode值是20（十六进制）。



还有两个地方需要注意。

首先，无论网页的原始编码是什么，一旦被Javascript编码，就都变为unicode字符。也就是说，Javascipt函数的输入和输出，默认都是Unicode字符。这一点对下面两个函数也适用。

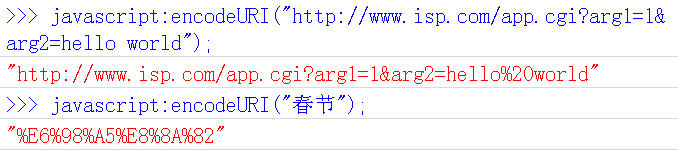


其次，escape()不对"+"编码。但是我们知道，网页在提交表单的时候，如果有空格，则会被转化为+字符。服务器处理数据的时候，会把+号处理成空格。所以，使用的时候要小心。

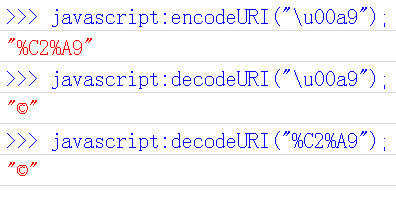
**七、Javascript函数：encodeURI()**

encodeURI()是Javascript中真正用来对URL编码的函数。

它着眼于对整个URL进行编码，因此除了常见的符号以外，对其他一些在网址中有特殊含义的符号"; / ? : @ & = + $ , #"，也不进行编码。编码后，它输出符号的utf-8形式，并且在每个字节前加上%。



它对应的解码函数是decodeURI()。

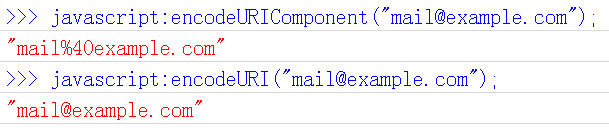


需要注意的是，它不对单引号'编码。

**八、Javascript函数：encodeURIComponent()**

最后一个Javascript编码函数是encodeURIComponent()。与encodeURI()的区别是，它用于对URL的组成部分进行个别编码，而不用于对整个URL进行编码。

因此，"; / ? : @ & = + $ , #"，这些在encodeURI()中不被编码的符号，在encodeURIComponent()中统统会被编码。至于具体的编码方法，两者是一样。



它对应的解码函数是decodeURIComponent()。