

文

让你完全理解base64是怎么回事 (/a/1190000004533485)

javascript (https://segmentfault.com/t/javascript/blogs)base64 (https://segmentfault.com/t/base64/blogs)张亚涛 (https://segmentfault.com/u/zhangyatao) 2 天前发布

HTTP将BASE64-编码用于基本认证和摘要认证，在几种HTTP扩展中也使用了该编码。

### Base-64编码保证了二进制数据的安全

Base-64编码可以将任意一组字节转换为较长的常见文本字符序列，从而可以合法地作为首部字段值。Base-64编码将用户输入或二进制数据，打包成一种安全格式，将其作为HTTP首部字段的值发送出去，而无须担心其中包含会破坏HTTP分析程序的冒号、换行符或二进制值。Base-64编码是作为MIME多媒体电子邮件标准的一部分开发的，这样MIME就可以在不同的合法电子邮件网关之间传输富文本和任意的二进制数据里。Base-64编码与将二进制数据文本化表示的uuencode和BinHex标准在本质上类似，但空间效率更高。

### 8位到16位

Base-64编码将一个8位字节序列拆散为6位的片段，并为每个6位的片短分配一个字符，这个字符是Base-64字母表中的64个字符之一。这64个输出字符都是很常见的，可以安全地放在HTTP首部字段中。这64个字符中包括 大小写字母、数字、+和／，还是用里特殊字符=。  
*注意：由于base64编码用了8位字符来表示信息中的6个位，所以base64编码字符串大约比原始值扩大了33%。*  
此处输入图片的描述

Base64 编码表							
Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

下面是一个简单的base64编码实例。在这里，三个字符组成的输入值“Owl”是base64编码的，得到的是4个字符的base64编码值“T3ch”。它是按以下方式工作的。

- (1) 字符串"Owl"被拆分成3个8位的字节(0x4F、0x77、0x21)。
- (2) 这3个字节构成了一个24为的二进制01001111 01110111 00100001。
- (3) 这些为被划分为一些6位的序列010011、110111、011100、1000001。
- (4) 每个6位值都表示了从0 ~ 63之间的数字，对应base64字母表中的64个字符之一。得到的base64编码字符串是4个字符的字符串“T3ch”。然后就可以通过线路将这个字符串作为“安全的”8位字符传送出去，因为只用了一些移植性最好的字符(字母、数字等)。

```
// 现在将字符串"Owl"转换为base64编码值
var str = 'Owl!';
// 或去字符串的二进制码
var binary = [];
for (var i = 0; i < str.length; i++) {
    // 转换为二进制表示
    var binStr = str.charCodeAt(i).toString(2);
    // 将得到的二进制放入数组中得到
    // ['1001111','1110111','100001']
    // 因为一个正常的二进制字节都是由8bit组成的, 不够8bit的话不表示.上面得到的都不够8bit所以前面我们手动给补0,就得到了
    // ['01001111','01110111','00100001']
    binary.push(binStr);
}

// 1 把字符串按照6位分开, 进行分割, 得到['010011','110111','011100','1000001']
// 2 将每一个转换为十进制分别对于 [19,55,28,33];
// 3 将每一位数字分别对于上面提供的base64对应表, 得到对应的编码, 分别对于  T 3 c h
// 4 最后就会得到base64编码T3ch
console.log('字符"Owl"最后得到的base64编码为"T3ch");
```

### base64填充

base64编码收到一个8位字节序列，将这个二进制序列流划分成6位的块。二进制序列有时不能正好平均地分为6位的块，在这种情况下，就在序列末尾填充零位，使二进制序列的长度成为24的倍数(6和8的最小公倍数)。  
对已填充的二进制进行编码时，任何完全填充(不包括原始数组中的位)的6位组都有特殊的第65个符号“=”表示。如果6位组是部分填充的，就将填充位设置为0。  
下面会写一个填充实例。初始输入字符串为"a:a"为3个字节(24位)。24是6和8的倍数，因此按照上面给出的例子计算。无需填充就会得到base64编码为"YTph"。  
然而，再增加一个字符，输入字符串变为"a:aa",转换为二进制就会有32位长。而6和8的下一个公倍数为48.因此要添加16为的填充码。填充的前4位是与数据位混合在一起的。得到的6位组01xxxx，会被当作010000、十进制中的16，或者base64编码的 Q 来处理。剩下的两个6位组都是填充码，用 = 来表示。

- a:a -- 011000 010011 101001 100001 -- YTph
- a:aa -- 011000 010011 101001 100001 011000 01xxxx xxxxxx xxxxxx -- YTphYQ==
- a:aa -- 011000 010011 101001 100001 011000 010110 0001xx xxxxxx -- YTphYWE=
- a:aaa -- 011000 010011 101001 100001 011000 010110 000101 1000001 -- YTphYWFh

最近的浏览器提供了自动生成base64的方法 `atob` 和 `btoa`

```
btoa('a:a')
// => "YTph"
atob('YTph')
// => "a:a"
```

希望此文可以帮助你完全理解BASE-64。

2 天前发布 (/a/1190000004533485)

0 推荐

收藏

你可能感兴趣的文章

Node.js 不深也不浅得了解下编码 (https://segmentfault.com/a/1190000002787763) 13 收藏, 1.5k 浏览

图片转 base64 编码显示 ( PHP ) (https://segmentfault.com/a/1190000002435029) 1 收藏, 1.1k 浏览

Base64开发者须知 (https://segmentfault.com/a/1190000003898830) 2 收藏, 158 浏览

本文采用 知识共享署名 3.0 中国大陆许可协议 (http://creativecommons.org/licenses/by/3.0/cn)，可自由转载、引用，但需署名作者且注明文章出处。

讨论区

不只33%，八位能存255个字符加个空字符。

(https://segmentfault.com/c/1050000004541433) **ranforce** (https://segmentfault.com/u/ranforce) · 1 天前

为什么一定要补位到 6 和 8 的公倍数，其实只要是 6 的倍数就好了，解码的时候 8 位 8 位的取，最后剩下的就肯定是补位的，撇掉不就好了么？

(https://segmentfault.com/c/1050000004548776) **公子** (https://segmentfault.com/u/lizheming) · 1 小时前

正常情况下是这样的。但是当连接两段BASE64编码过的字符串后再解码的时候，这个时候就需要6和8的公倍数了

(https://segmentfault.com/c/1050000004548900) **pixelstech** (https://segmentfault.com/u/pixelstech) · 1 小时前

回复 pixelstech (https://segmentfault.com/u/pixelstech):  
哇靠...还能对 base64 进行拼接！逆天了😏


(https://segmentfault.com/c/1050000004548921) **公子** (https://segmentfault.com/u/lizheming) · 1 小时前

请先 登录 () 后评论

本文隶属于专栏

张亚涛 (https://segmentfault.com/blog/zhangyatao)

个人的技术总结

 张亚涛 (https://segmentfault.com/u/zhangyatao)  
作者

关注专栏

分享扩散：  
...