**子页面中调用父页面的函数，或者其他页面调用其他页面的函数，被调用者要先被加载。**

**子页面需要用到的jquery最好都放到父页面中，子页面重复引用可能会发生冲突。**

**数据库中字段的的true/false可能是字符串，虽然看着是true和false，但布尔值和字符串是两种数据类型，js中不可以if (这样判断) {}，可能是一个坑。**

**第三方组件使用前要初始化！**

**区别正则中?、+、\*三个字符区别**

**?:出现一次或零次，最多一次**

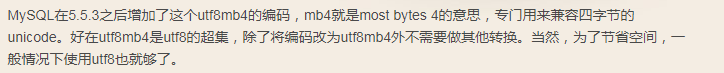
**+:出现一次或多次，最少出现一次**

**\*:出现零次或多次，可以不出现**

**return配合三元运算符**

**return a > b ? return true : return false;**

**Utf8和utf8mb4的区别**



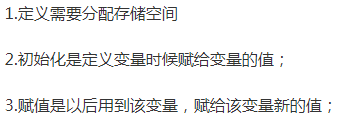
·区别之一就是前者节省空间，后者兼容性好。Uf8mb4的应用场景之一时保存emoji表情。

**定义、初始化、赋值**

**String parame; // 分配内存，但内存没东西**

**String parame2 = “value”; // 分配内存的同时赋予东西**

**parame = parame2; // 更改内存的东西**



[**https://zhidao.baidu.com/question/108348358.html**](https://zhidao.baidu.com/question/108348358.html)

**区分有30天和31天的月份**

**一三五七八十腊，三十一天永不差。四六九冬三十天，平年二月二十八**

Chrome调试快捷键

F8：恢复脚本执行

F10：完成下一个函数调用（单步执行，不进入调用函数）！不进入方法

F11：进入下一个函数调用（单步执行，进入调用函数）！进入方法

在代码中加入js的debugger关键字即可调试

eclipse调试快捷键

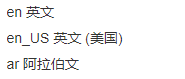
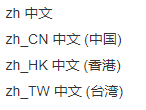
F5：进入方法

F6：下一行

F7：跳出该方法

F8：恢复执行

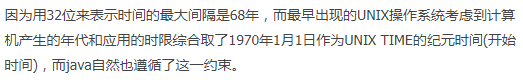
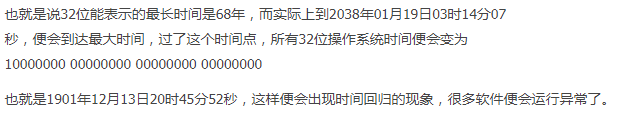
[chrome快捷键大全](https://blog.csdn.net/hustxiayu/article/details/52055832)



<https://blog.csdn.net/fenglailea/article/details/45888799>

[格林威治时间](https://www.cnblogs.com/FrankLei/p/6815303.html)

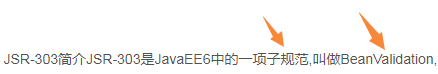
[为什么是1970.01.01](https://www.cnblogs.com/FrankLei/p/6815303.html)



什么是快照?

快照是一种特殊的版本，指定了某个当前的开发进度的【副本】。不同于常规的版本，Maven 每次构建都会在远程仓库中检查新的快照。 现在 data-service 团队会每次发布更新代码的快照到仓库中，比如说 data-service:1.0-SNAPSHOT 来替代旧的快照 jar 包。

JSR-303



用这规范的规则来校验Bean的值，可用注解来进行校验

<https://www.aliyun.com/jiaocheng/778566.html?spm=5176.100033.2.5.392742abLgMrwi>

eclipse中查看java源码



<https://zhidao.baidu.com/question/548257856.html>

关闭端口，dos中操作



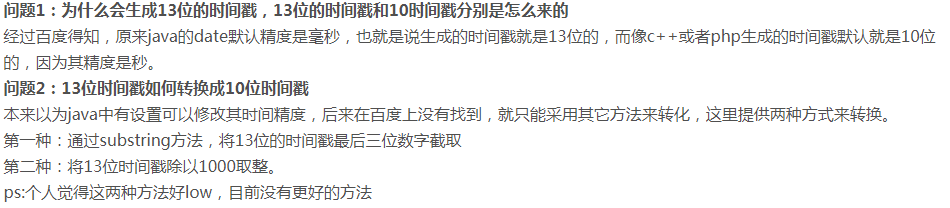
得到进程号之后

·在任务管理器中结束进程

·在Dos中结束进程(tskill PID)

<https://blog.csdn.net/lll921176681/article/details/76618336>

十三位的时间戳和十位的时间戳



<https://blog.csdn.net/qq_28483283/article/details/80583197>

src = source      源  
dst = destination 目的

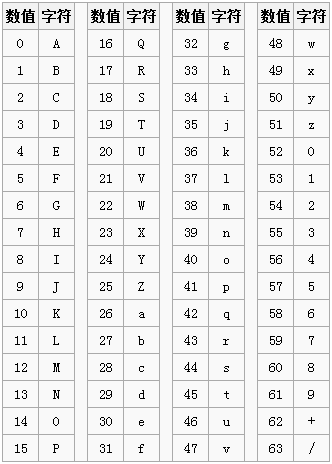
base64

一种用64个字符来表示任意二进制数据的方法。

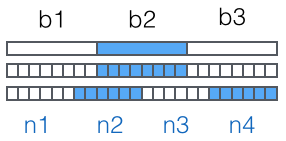
为什么会有Base64编码呢？

因为有些网络传送渠道**并不支持所有**的字节，例如传统的邮件只支持可见字符的传送，像ASCII码的控制字符就不能通过邮件传送。这样用途就受到了很大的限制，比如图片二进制流的每个字节不可能全部是可见字符，所以就传送不了。最好的方法就是在不改变传统协议的情 况下，做一种扩展方案来支持二进制文件的传送。**把不可打印的字符也能用可打印字符来表示，问题就解决了**。Base64编码应运而生，Base64就是一种基于64个可打印字符来表示二进制数据的表示方法。

**Base64索引表**



Base64的编码只有6个bit表示，正常的字符是8个bit表示，6和8的最小公倍数24，那么就是4个base64字符可以表示3个标准ascii字符。



例子：

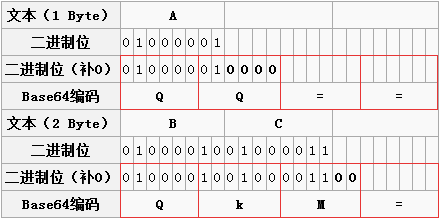
1、把abc三个字符转换为base64

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字符串 | a | b | c |  |
| ASCII | 97 | 98 | 99 |  |
| 8bit | 01100001 | 01100010 | 01100011 |  |
| 6bit | 011000 | 010110 | 001001 | 100011 |
| 十进制 | 24 | 22 | 9 | 35 |
| 对应base64编码 | Y | W | J | j |

2、把man三个字符转换为base64

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字符串 | m | a | n |  |
| ASCII | 109 | 97 | 110 |  |
| 8bit | 01101101 | 01100001 | 01101110 |  |
| 6bit | 011011 | 010110 | 000101 | 101110 |
| 十进制 | 27 | 22 | 5 | 46 |
| 对应base64编码 | b | W | F | u |

3、当转换到最后的字符不足3个字符咋办，如果不足三个字符的话，我们直接在最后添加“＝”号即可（左边开始）



但是由于 “=” 字符会在URL和Cookie中引起歧义，所以很多Base64编码会把后边的“=”字符去掉。那么如果去掉了岂不是破坏了完整性，到时怎么解码？因为Base64编码的长度永远都是4的倍数，所以保证Base64编码的长度是4的倍数就可以正常解码了（加“=”字符）。

套路总结：

字符转base64，找ascii对照表 - 换二进制 - 位数切换 - 找base64对照表

base64转字符，找base64对照表 - 换二进制 - 位数切换 - 找ascii对照表

其中

转base64时，二进制位数不够用‘0’来凑

转字符时，base64码长度不够用‘=’来凑（4的倍数）

[Base64原理解析](https://www.cnblogs.com/diligenceday/p/6002382.html)

[base64](https://www.liaoxuefeng.com/wiki/001374738125095c955c1e6d8bb493182103fac9270762a000/001399413803339f4bbda5c01fc479cbea98b1387390748000)

**Uri代表要操作的数据，Android上可用的每种资源 - 图像、视频片段等都可以用Uri来表示。**

**Android的Uri由以下三部分组成： "content://"、数据的路径、标示ID(可选)**

**如：**

**所有联系人的Uri： content://contacts/people**

**某个联系人的Uri: content://contacts/people/5**

**所有图片Uri: content://media/external**

**某个图片的Uri：content://media/external/images/media/4**

# [Android中Uri的使用](https://www.cnblogs.com/JLZT1223/p/6805558.html)

字节与byte的关系

1字节= 8位（1 byte = 8 bit）

**1 Byte = 8 Bits**

**1 KB = 1024 Bytes = 2 ^ 10 bytes**

**1 MB = 1024 KB = 2 ^ 20 bytes = 1024 \* 1024**

**1 GB = 1024 MB = 2 ^ 30 bytes = 1024 \* 1024 \* 1024**

[**字节、字、bit、byte的关系**](https://www.cnblogs.com/chansblogs/p/6542215.html)

Android中变量命名规范

非公有变量前面要加上小写m

private String mtestVariable;

String mtestVariable;

protected String mtestVariable;

静态变量全大写，下划线分割

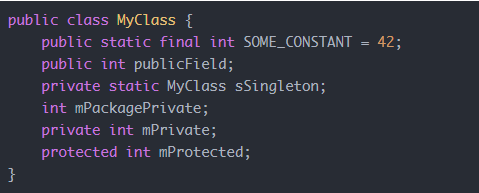
public static final int SOME\_CONSTANT = 42;

静态非公有变量前面加上小写s

private static MyClass sSingleton;

其它变量以小写字母开头

public int publicField;



# [关于变量名前面加m的问题](https://blog.csdn.net/ameryzhu/article/details/12236215)

Android中VISIBLE、INVISIBLE、GONE的区别

VISIBLE：显示控件，控件可点击（若控件有点击事件）

INVISIBLE：不显示控件，但保留控件所占有的空间，控件不可点击（若控件有点击事件）

GONE：隐藏控件，即不保留控件所占有的空间，控件不可点击（若控件有点击事件）

#### 代码写法（java代码中）：

view.setVisibility(View.VISIBLE);

view.setVisibility(View.INVISIBLE);

view.setVisibility(View.GONE);

#### 代码写法（XML布局文件中）：

android:visibility="visible"

android:visibility="invisible"

android:visibility="gone"

# [Android中VISIBLE、INVISIBLE、GONE的区别](https://blog.csdn.net/u010005281/article/details/78739957)

[MIME(多用途互联网邮件拓展类型)](https://baike.baidu.com/item/MIME/2900607?fr=aladdin)

MIME(Multipurpose Internet Mail Extensions)多用途互联网邮件扩展类型。**是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。**多用于指定一些客户端自定义的文件名，以及一些媒体文件打开方式。

它是一个互联网标准，扩展了电子邮件标准，使其能够支持：

多用途互联网邮件扩展，它是一个互联网标准，**在1992年最早应用于电子邮件系统，但后来也应用到浏览器。**服务器会将它们发送的多媒体数据的类型告诉浏览器，而通知手段就是说明该多媒体数据的MIME类型，从而让浏览器知道接收到的信息哪些是MP3文件，哪些是Shockwave文件等等。服务器将MIME标志符放入传送的数据中来告诉浏览器使用哪种插件读取相关文件。

**最早的HTTP协议中，并没有附加的数据类型信息，所有传送的数据都被客户程序解释为超文本标记语言HTML 文档**，而为了支持多媒体数据类型，HTTP协议中就使用了附加在文档之前的MIME数据类型信息来标识数据类型。

MIME意为多功能Internet邮件扩展，它设计的最初目的是为了在发送电子邮件时附加多媒体数据，让邮件客户程序能根据其类型进行处理。然而当它被HTTP协议支持之后，它的意义就更为显著了。**它使得HTTP传输的不仅是普通的文本，而变得丰富多彩。**

每个MIME类型由**两部分组成**，前面是**数据的大类别**，例如声音audio、图像image等，后面定义**具体的种类**。

MIME头**根据在邮件包中的位置**，大体上分为**MIME信息头**和**MIME段头(MIME信息头指整个邮件的头，而MIME段头只每个MIME段的头)**。在一个符合MIME的信息中，也包含一个信息头并不奇怪，邮件的各个部分叫做MIME段，每段前也缀以一个特别的头。MIME邮件只是基于RFC 822邮件的一个扩展，然而它有着自己的RFC规范集。

MIME信息头有：

MIME-Version:

这个头提供了所用MIME的版本号。这个值习惯上为1.0。

Content-Type:

它定义了数据的类型，以便数据能被适当的处理。有效的类型有：text，image，audio，video，applications，multipart和message。注意任何一个二进制附件都应该被叫做application/octet- stream。这个头的一些用例为：image/jpg, application/mswork，multipart/mixed，这只是很少的一部分。

Content-Transfer-Encoding:

这是所有头中最重要的一个，因为它说明了对数据所执行的编码方式，客户/MUA 将用它对附件进行解码。对于每个附件，可以使用7bit，8bit，binary ，quoted-printable，base64和custom中的一种编码方式。7bit编码是用在US ASCII字符集上的常用的一种编码方式，也就是，保持它的原样。8bit和binary编码一般不用。对人类可读的标准文本，如果传输要经过对格式有影响的网关时对其进行保护，可以使用quoted printable 。Base64是一种通用方法，在需要决定使用哪一种编码方法时，它提供了一个不用费脑子的选择；它通常用在二进制，非文本数据上。注意，任何非7bit 数据必须用一种模式编码，这样它就可以通过Internet邮件网关！

Content-ID:

如果Content-Type是message/external-body或multipart/alternative时，这个头就有用了，它超出了本文的范围。

Content-Description:

这是一个可选的头。它是任何信息段内容的自由文本描述。**描述必须使用us-ascii码**。

Content-Disposition:

一个试验性的头，它用于给客户程序/MUA提供提示，来决定是否在行内显示附件或作为单独的附件。

MIME段头（出现在实际的MIME附件部分的头），除了MIME-Version头，可以拥有以上任何头字段。如果一个MIME头是信息块的一部分，它将作用于整个信息体。例如，如果Content-Transfer-Encoding显示在信息（指整个信息）头中，它应用于整个信息体，但是如果它显示在一个MIME段里，它"只能"用于那个段中。

注意：其可以对自动对收到的邮件进行解密。