**图片轮番**

**CSS cursor 属性：**

pointer光标呈现为指示链接的指针（一只手）

move此光标指示某对象可被移动。

text此光标指示文本。

wait此光标指示程序正忙（通常是一只表或沙漏）。

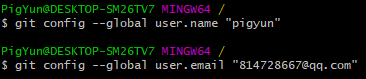
help此光标指示可用的帮助（通常是一个问号或一个气球）。

**display：block；**比较常用于<a><span>这两个标签——因为这两个标签非块元素，如果不用display：block定义一下，那么定义width、height等和[长宽](https://www.baidu.com/s?wd=%E9%95%BF%E5%AE%BD&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9nyu9nj7WrycsPhP9mW9B0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHb4nW63nH64" \t "_blank)相关的css属性时会发现完全不生效。

**z-index**取值12345时候，保证有几个position：absote的时候不会被覆盖！！！！很重要！！！！！

**Git教程**

Git是目前世界上最先进的分布式版本控制系统（没有之一）



$git config命令的--global参数，用了这个参数，表示你这台机器上所有的Git仓库都会使用这个配置

$ mkdir learngit

$ cd learngit

$ pwd

$ git init 把这个目录变成Git可以管理的仓库

$ git add readme.txt

$ git commit -m "wrote a readme file"

$ git status 让我们时刻掌握仓库当前的状态

$ git diff readme.txt git diff顾名思义就是查看difference

$ git log 显示从最近到最远的提交日志

HEAD指向的版本就是当前版本，上一个版本就是HEAD^，上上一个版本就是HEAD^^,因此，Git允许我们在版本的历史之间穿梭

$ git reset --hard commit\_id。穿梭前，用$ git log可以查看提交历史，以便确定要回退到哪个版本。版本回退

$ git reflog查看命令历史，以便确定要回到未来的哪个版本。

$ git checkout -- file 把readme.txt文件在工作区的修改全部撤销

$ git reset HEAD readme.txt 把暂存区的修改回退到工作区

$ git rm 从版本库中删除该文件，并且git commit

$ git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

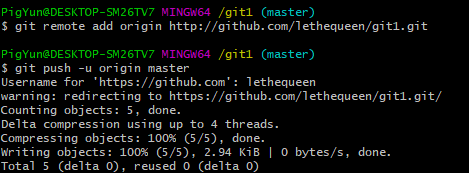
**远程操作**

$ ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com) 在本地得到公匙

$ git remote add origin [git@server-name:path/repo-name.git](mailto:git@server-name:path/repo-name.git) 要关联一个远程库

这里报错得话用 git remote rm origin

$ git push -u origin master 第一次推送master分支的所有内容, 此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改

****

$ git clone [git@github.com:michaelliao/gitskills.git](mailto:git@github.com:michaelliao/gitskills.git) 要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆

**分支管理**

$ git checkout -b <name> git checkout命令加上-b参数表示创建并切换，相当于以下两条命令：

$ git branch <name> 创建分支

$ git branch 查看分支

$ git checkout <name> 切换分支

$ git merge <name> 用于合并指定分支到当前分支

$ git branch -d <name> 删除分支

$git log –graph 看到分支合并图

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

$ git merge --no-ff -m "merge with no-ff" dev 合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

在实际开发中，每个开发在公司远程仓库，都有一个对应每个人自己的远程分支，每次提交时教程里面讲git push 中master参数换成对应开发人员的分支就好了，比如devloper1，控制版本上线的人比如运维人员，在远程仓库的服务器上，再选择性地把某个开发人员的分支合并进真正的master实现上线

$ git stash 可以把当前工作现场“储藏”起来，等以后恢复现场后继续工作

当手头工作没有完成时，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场。

$ git branch -D <name> 如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除。

$ git remote 查看远程库的信息, 或者，用git remote -v显示更详细的信息：

**多人协作的工作模式通常是这样**：

首先，可以试图用git push origin branch-name推送自己的修改；

如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；

如果合并有冲突，则解决冲突，并在本地提交；

没有冲突或者解决掉冲突后，再用git push origin branch-name推送就能成功！

如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream branch-name origin/branch-name。

这就是多人协作的工作模式，一旦熟悉了，就非常简单。



$ git tag <name> 新建一个新标签

$ git tag 查看所有标签

$ git show <tagname> 查看标签信息

$ git tag -a v0.1 -m "version 0.1 released" 3628164 用-a指定标签名，-m指定说明文字

$ git tag -s <tagname> -m "blablabla..." 可以用PGP签名标签；

$ git push origin <tagname> 可以推送一个本地标签

$ git push origin –tags 可以推送全部未推送过的本地标签；

$ git tag -d <tagname> 可以删除一个本地标签；

$ git push origin :refs/tags/<tagname> 可以删除一个远程标签。

目前为止，我所熟知的左中右三栏宽度自适应于浏览器的方法有三个：[绝对定位法](http://www.zhangxinxu.com/wordpress/2009/11/%E6%88%91%E7%86%9F%E7%9F%A5%E7%9A%84%E4%B8%89%E7%A7%8D%E4%B8%89%E6%A0%8F%E7%BD%91%E9%A1%B5%E5%AE%BD%E5%BA%A6%E8%87%AA%E9%80%82%E5%BA%94%E5%B8%83%E5%B1%80%E6%96%B9%E6%B3%95/#m1)，[margin负值法](http://www.zhangxinxu.com/wordpress/2009/11/%E6%88%91%E7%86%9F%E7%9F%A5%E7%9A%84%E4%B8%89%E7%A7%8D%E4%B8%89%E6%A0%8F%E7%BD%91%E9%A1%B5%E5%AE%BD%E5%BA%A6%E8%87%AA%E9%80%82%E5%BA%94%E5%B8%83%E5%B1%80%E6%96%B9%E6%B3%95/#m2)以及[自身浮动法](http://www.zhangxinxu.com/wordpress/2009/11/%E6%88%91%E7%86%9F%E7%9F%A5%E7%9A%84%E4%B8%89%E7%A7%8D%E4%B8%89%E6%A0%8F%E7%BD%91%E9%A1%B5%E5%AE%BD%E5%BA%A6%E8%87%AA%E9%80%82%E5%BA%94%E5%B8%83%E5%B1%80%E6%96%B9%E6%B3%95/#m3)。这些方法简洁实用，且无兼容性问题。