在互联网行业工作的想必都多多少少听说过GitHub的大名，除了是最大的开源项目托管平台，许多企业也都是用GitHub来协同开发工作，当然我们彩程也是其中之一。笔者最初决定学习Git也是因为在团队内部设计方案初步被开发出来后，难免会有一些细节需要调整，而为了调整几像素的问题再求前端工程师出马，其实是很影响整体效率的，所以希望通过学习GitHub好在必要的时候直接参与开发，能发挥自己的一点CSS技术以更直接快速的解决问题。

但是期间竟然发现网上无一篇为设计师而备的Git教程，典型的，几乎没有一个教程讲了GitHub的官方应用，而都是围绕命令符做教程，对就是DOS那样的命令符界面，这对于设计师和产品经理等非技术背景从业者来说多少有些难于上手，所以笔者打算结合个人体验，自己动手写一篇以使用网页和客户端为主的GitHub教程（暂时先仅以Mac版客户端为例），希望它能帮助大家以最快速度熟悉并逐渐开始使用GitHub。

**为什么要了解GitHub**

在开始之前，还是得说说为什么GitHub值得设计师学习。除了开篇所讲的例子，可以将自己所了解的一点CSS或开发技能发挥出来，降低团队合作中的沟通成本。这本质上还是设计师是否应该学习开发或学习到什么程度的问题。不过我一直认为这没什么可探讨的，只是每个人不同的选择和追求。就像有些人是想成为家具设计师，有些人是真想亲手打磨出自己心中最理想的那一把椅子；有些人想成为建筑设计师，有些人只想在山脚下为家人盖一座完美小屋。同样，如果你是真的热爱这个日新月异的行业，没准某天也要自己创造点什么，我想你一定会毫不犹豫的开始学习。

**什么是Github**

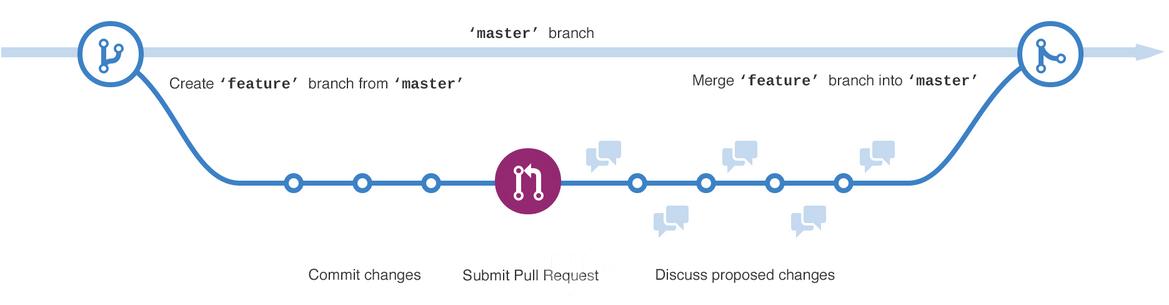
按惯例，还是先简介下究竟什么是Git（有基本认识的同学可以直接跳过）。让我们摒弃那些专业名词，用设计师工作中经常遇到的情况来介绍什么是Git：

* 你出了一版方案A，在大家讨论后，你改到B，结果再讨论，大家觉得还是不如用A，这时发现A没有另存，还得再改回去！在GitHub中就不用担心，它会记录你提交的每个版本，并把这些都放在一个仓库（Repository）里，而每一次提交改变就是 Commit，你可以随时回退到任意一个版本。
* 此外你还很可能遇到方案衍生的情况，在方案A的基础上，改出了一种方案B，又改出了一种方案C，可能还分别衍生出B1、B2及C1、C2，在GitHub里有分支（Branches）可以记录这种方案的分化过程。
* 后来你觉得B1和C2方案中都有可取的地方，把它们融合一下就可以输出终稿了，Git里当然也支持这种分支合并（Merged）。



当你个人使用Git，涉及的基本概念就这么几条，是不非常简单呢。下面我们来看看多人协同，也就是Git真正厉害的地方，当然也不复杂。

* 在稍大的团队中，可能需要几个设计师合作完成一个方案，怎么样统一进度呢，一种就是大家每天把文件拷在一起，这需要经常浪费时间去同步文件，显然很不方便。另一种是每个人电脑里都留一份，需要时就和云端服务器同步，Git就是采用这样的所谓分布式系统。好处是更安全，也更便捷。
* 那么问题也就来了，如果大家都改同一个东西，万一冲突了怎么办？不用担心，Git会帮你对比并告诉你哪里有冲突了，你可以逐个对冲突的地方做出抉择。此外，前面不是讲到可以衍生出分支（Branches）么，在大家各自推进方案的时候，分别把内容放在不同的分支里，就不会相互干扰了。
* 开源的项目是任何人都可见，你可以Fork一个项目，这相当于在你的账号下从原项目新建了一个分支，你可以在此基础上改动，如果有希望提交给原作者的成果，可以发合并申请到原库（Pull Request），原作者可以看到通知并决定是否合并。通过这种方式，大家就可以合力将某个开源项目变得更好。



最后我们还得明确一下概念，Git和GitHub有什么区别，引用知乎上[Fluyy](http://www.zhihu.com/question/21907548)的解释“git是一个版本控制工具，github是一个用git做版本控制的项目托管平台。”这有点类似于Wordpress和Wordpress.com的关系，前者是一个任何有都可以用的免费博客系统，后者是一个平台，在这个平台上你可以通过注册来直接使用Wordpress写博客。

说到这里我想你已经对GitHub是怎么回事有了基本的认识，下面就让我们撸起袖子，以一个简单的案例来看看GitHub到底是怎么使用吧。

**案例学习-做一遍全明白！**

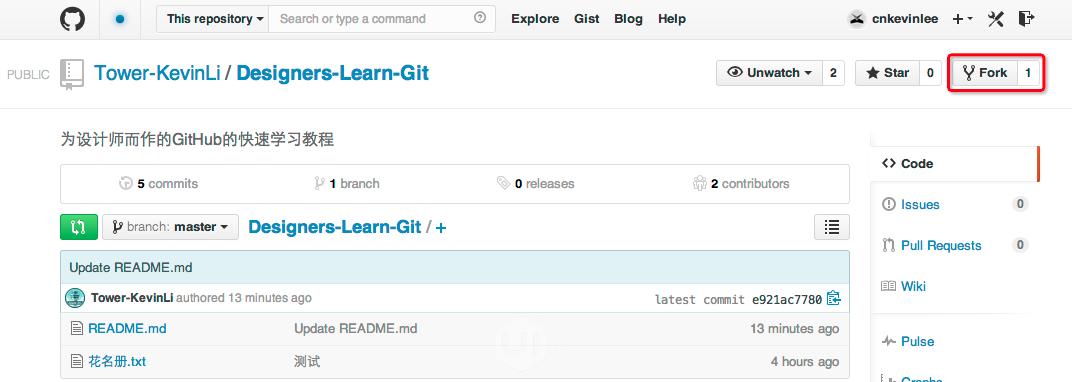
我们以Fork一个项目，做出自己的修改并提交给原作者的任务作为基本案例，下面跟着我一步一步来吧，网快手快的同学几分钟就可以搞定啦。

**第一步：注册一个GitHub账号并登录**

这个流程就不说了，完成后可以[下载GitHub官方App](https://mac.github.com/)，并登录。

**第二步：Fork仓库**

[点这里](https://github.com/Tower-KevinLi/Designers-Learn-Git)打开我用马甲专为本教程建立一个库 Designers-Learn-Git，可以看到创建者名（我的马甲）写在/前面：Tower-KevinLi。然后点击右上角的Fork按钮。



完成后你会发现跳到一个新的页面，Tower-KevinLi变成了你的GitHub账户名（cnkevinlee是我的另一个马甲...），这表示你的账号下已经“复制了”一份Designer-Learn-Git，然后你就可以做修改了。



**第三步：修改文件**

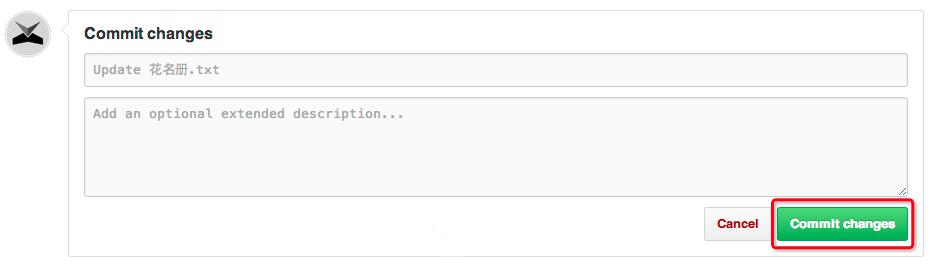
可以看到Designers-Learn-Git这个仓库里只有两个文件“README.md”（项目说明）和“花名册.txt"，这里我们只需尝试修改后者（道理和修改有很多文件的仓库是一样的）。可以直接在线修改，也可以先克隆到本机再修改，对于比较复杂的项目肯定是要采取后者，不过这里我们可以先看看在网页上直接修改怎么操作。

**在线修改**：直接在页面上点”花名册.txt“的名字，进入如下详情页，再点击”Edit“。

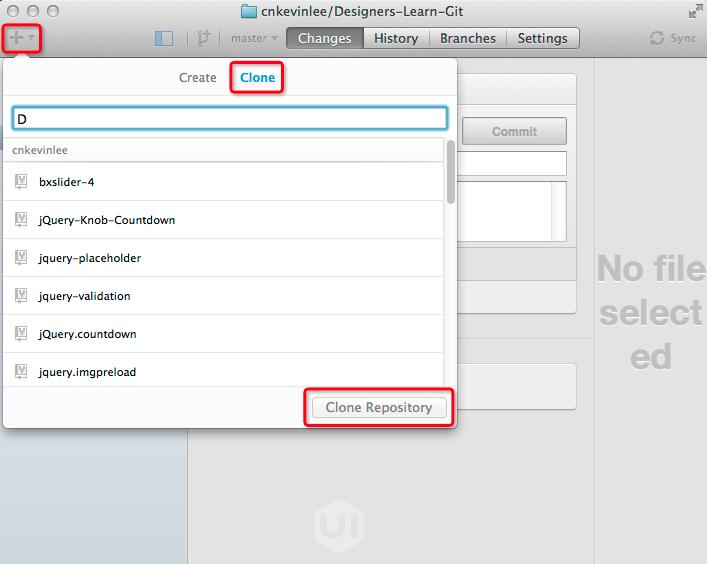


这里设计的任务内容是在花名册上随便写写你的ID和Tiitle，初衷是能通过这个和学习这个教程的同学有一个互动，看看你能排到第几位。

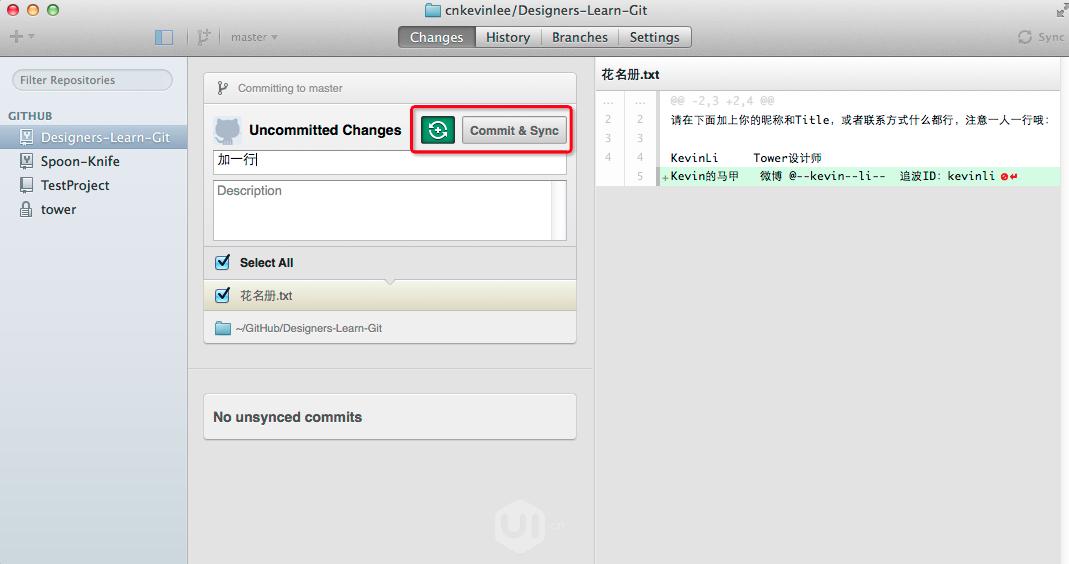
在编辑页面编辑完后，滚动到页面底部，点击绿色的“Commit Changes”按钮确认提交。



**本地修改**：另一种办法是把项目克隆到本地后再修改，打开客户端（这里以Mac最新版为例），点击右上角的“+”号，切换到“Clone”，找到“Designers-Learn-Git”后点击右下的“Clone Repository”按钮。



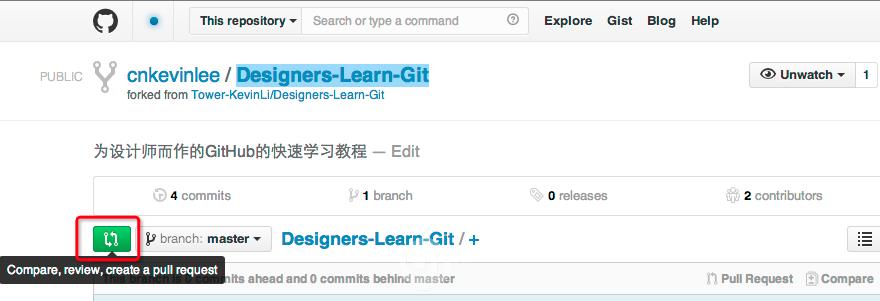
然后从本地找到克隆的文件夹，打开“花名册.txt”，编辑并保存。这时再切换到GitHub应用的窗口，你会发现它在“Changes”页已经检测到并列出了你的改变，然后点击Commit & Sync按钮，将修改提交并同步到GitHub。注意，提交和同步是两个动作，需要先将下图那个绿色的按钮激活，这两个动作才会同时执行，否则你就需要在提交后再点整个界面右上角的Sync（适用于做出多个修改后统一提交的情况）。



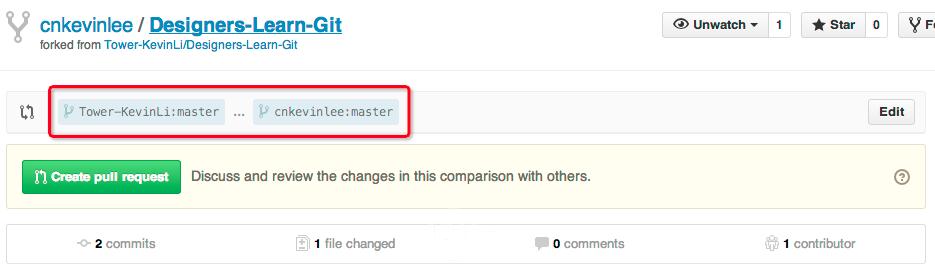
**第四步：申请合并**

在确认已经对自己Fork了的仓库修改成功后，你可以将提交合并申请，申请将你的版本合并入最初的项目也就是我创建的 Designers-Learn-Git，步骤如下：

在网页上打开你Fork的Designers-Learn-Git的页面，点击这个醒目的绿色按钮



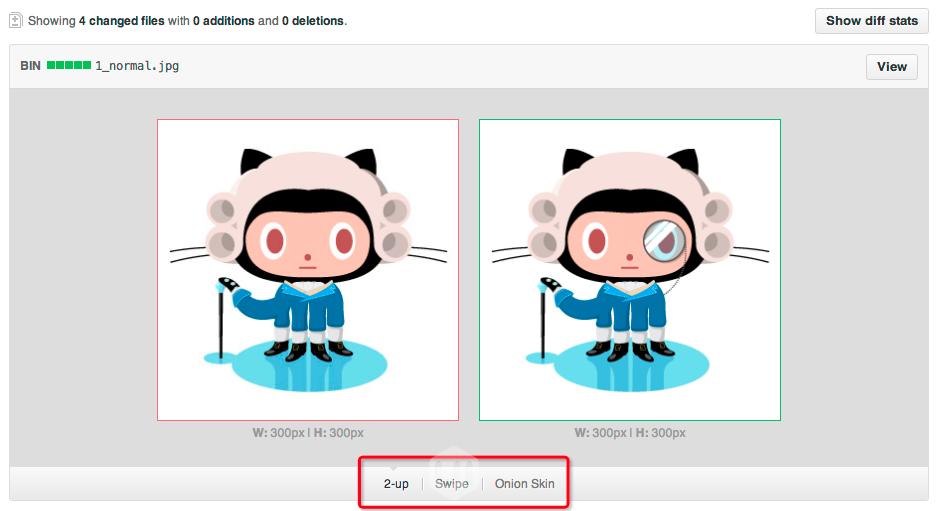
再点击“Create pull request”按钮，提交申请，完成后我会收到通知，并将你补充的内容合并入原库。



注意上图圈红的地方显示了要合并的两个分支，由于没有新建分支，这里就默认显示两个库都仅有的Master主分支，你可以尝试新建分支，并选择其它分支申请合并，体验GitHub的分支功能，本文作为入门教程这里就先不赘述了，有问题的同学可以联系我。

**图片对比功能，设计师的真爱！**

做完前面的案例，你会发现GitHub可以直接对比文件间的改动，但也仅对程序、文本文件有效，可我们设计师都是靠图吃饭啊！好了别急，GitHub一直在努力提升设计师的使用体验，并且已经实现了很好的图片对比功能，当然直接上传可能巨大的PSD源文件不太现实，但我们可以对比导出后的版本，[点击这里看看Demo](https://github.com/cameronmcefee/Image-Diff-View-Modes/commit/8e95f70c9c47168305970e91021072673d7cdad8)吧。

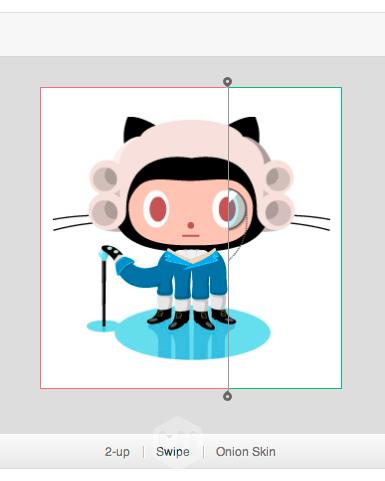


GitHub的图片对比工具提供了三种对比方法帮你找不同，我们来挨个看一下：

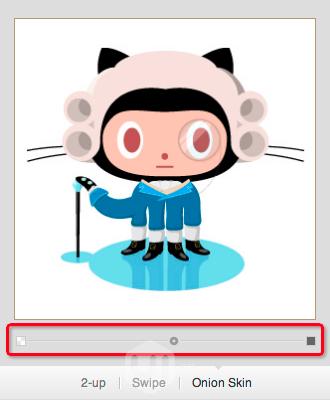
**2-up**：就是直接并排放在一起对比，会显示尺寸：



**Swipe**：是将两个图摞在一起，通过拖动，改变上面的图的显示大小，用户盯着拖动线附近的变化就可以快速发现区别了，如图，当拖到猫眼附近就很容易看出一个戴了眼镜。



**Onion Skin**：也是将两个图摞在一块，图下方有一个拖动条，控制上面一张图的透明度，这样在拖动改变透明度的时候，就能感觉到有区别的地方了。



**GitHub使用拓展-能做的远不止这些！**

版本管理，协同开发与设计，这只是GitHub最基本的用途，事实上利用Git的体系，可以实现很多其它事情。

**用GitHub协作翻译**

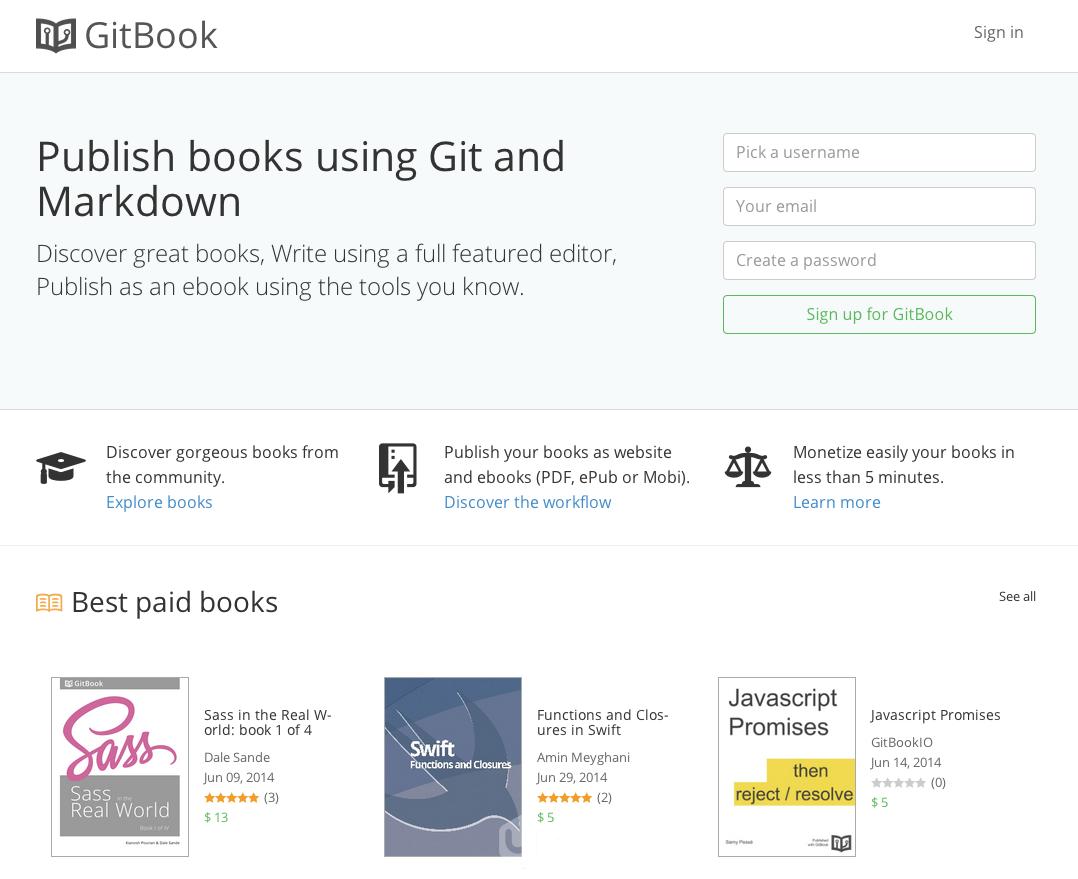
苹果发布Swift语言，号称更加容易上手，让很多设计师都跃跃欲试。官方同时发布了（翻译成中文后）达300多页的官方手册，而国内一个自发组织起来的团队，30多个人用9天时间即将翻译和校对工作全部完成，他们每人都还有自己的事情，上班、上线、创业，然而借助GitHub他们仅用业余时间在这么短的时间内就完成了这一壮举。他们的口号是“这一次，让中国和世界同步”



[点这里](https://github.com/numbbbbb/the-swift-programming-language-in-chinese)可以看到他们的项目和翻译成果。

**用GitHub写书**

说到了前一个例子，就不得不提[GitBook](https://www.gitbook.io/)，一个结合GitHub和Markdown来制作精美在线读物的工具。你可以自己或和任何人合作，编著一本在线书籍，还能够销售，当作家不再只是梦。



**用GitHub进行项目管理**

GitHub最初是为了开发的管理而生，当然也就具备了项目管理的潜质，特别是与开发密切联系的项目中，它的优势尽显。这几篇文章介绍了如何使用GitHub结合其它工具进行项目管理：《[Using Github for Project Management](http://liftux.com/posts/using-github-issues-project-management/)》、《[使用GitHub进行团队合作](http://xiaocong.github.io/blog/2013/03/20/team-collaboration-with-github/)》。当然，GitHub还是很偏重开发的管理，一般的项目管理还是适合使用我们家[Tower](https://tower.im/)之类的产品 ：）

**用GitHub搭建博客、个人网站**

一个在线的个人页面，独立博客，几乎是每个设计师的必备。过去，要自己租空间、安装网站程序，搭个人网站，是个耗时又好钱的事。而GitHub本身提供免费的托管服务，又提供了贴心的Pages功能，可以绑定你自己的域名，让这一切就水到渠成了。免费、高效、不限流量，做一个个人页面绰绰有余，比如前两天碰巧看到的这个例子：[jianglai.me](http://jianglai.me/)。不过其实很多工程师都已经在GitHub上搭建了自己的博客，设计师们也快跟进吧。

推荐一些相关教程：

* 《[通过GitHub Pages建立个人站点（详细步骤）](http://www.cnblogs.com/purediy/archive/2013/03/07/2948892.html)》
* 《[如何搭建一个独立博客——简明Github Pages与Hexo教程](http://jianshu.io/p/05289a4bc8b2)》
* 《[搭建一个免费的，无限流量的Blog----github Pages和Jekyll入门](http://www.ruanyifeng.com/blog/2012/08/blogging_with_jekyll.html)》

**更多进阶教程推荐**

鉴于本文只是一篇抛砖引玉的入门教程，就不再详述更多的细节，如果对Git感兴趣的同学，可以看看下面这些，它们可以帮助你成为一名专家、至少是设计师中的Git专家 :)

* 《[git-简明指南](http://rogerdudler.github.io/git-guide/index.zh.html)》
* 《[Pro Git](http://git-scm.com/book/zh)》
* 《[Git Magic](http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/zh_cn/index.html)》
* [一个学习Git的在线互动教程](http://pcottle.github.io/learnGitBranching/)
* [知乎上关于GitHub学习资料的问题](http://www.zhihu.com/question/20070065)

当然这也许不是最好的GitHub入门教程，但一定是最适合设计师的，希望它能帮助你快速的入门了解这一程序猿们整天挂在嘴边的东西。文中如有不恰当之处请指出，有不清楚的地方也欢迎联系我 [@彩程-Kevin](http://weibo.com/1966758563) 讨论。

当然，GitHub只是技术世界的一个小支脉，问题的根本，还是设计师对于技术的学习，如果你是个不满足于绘图工具，想做Geek的设计师，欢迎加群 302258924 交流。

【编辑推荐】

1. [被控性别歧视，GitHub联合创始人辞职](http://developer.51cto.com/art/201404/436805.htm)
2. [GitHub上最流行的10000个Java都使用了哪些库？](http://developer.51cto.com/art/201404/437074.htm)
3. [程序员必读：七款出色的GitHub功能集成工具](http://developer.51cto.com/art/201405/439242.htm)
4. [ASP.NET入驻Github 下一代ASP.NET将全部开源](http://developer.51cto.com/art/201405/439491.htm)
5. [为网页设计师和开发者准备的20个很棒的JavaScript资源](http://developer.51cto.com/art/201406/442720.htm)