Welcome to the Developers Corner

This is our guide for teaching you how to join the open-source mobile developer community, and build apps that are better because they are open, transparent, more secure, more safe, well designed and all together awesome!

Content Sources and Translations

This content is pulled from many different locations. Also, some of this content will be in languages other than english. If the translation needs to be improved, please perform a pull request.

Table of Contents

We have broken down content into something easily understandable for anyone new to open source, civic hacking, and mobile development.

- Taking smart photos
- Team and Project Management
- Building an open source community
- How to use IRC (Internet Relay Chat)
- The story of self and motivating a movement
    - TRAINERS NOTES: STORY OF SELF
- Top five open source project management tools
- Five ways to boost community engagement # A BEGINNER’S GUIDE TO CONTRIBUTING TO OPEN SOURCE

By Natasha Murashev

Source General Assembly Blog

Natasha is a iOS Engineer at Capital One Labs and instructor at GA in San Francisco. This post originally appeared on her blog, Natasha The Robot.

When I first started as a Rails developer, I thought of all the Rails gems as magic. Some smart people were making all these cool libraries for me to use! I had no idea how any of these libraries worked, and I was ok with that. They worked and did what I needed them to do. They seemed so comprehensive and thought out, that I didn’t even know what I would contribute to them even if I wanted to!

To this day, I haven’t contributed any open source code to the Rails community. And that’s because the Rails community is extremely active on open source, so it’s actually hard to find things to contribute to! Of course you can go through issues and try to solve them, but they are usually pretty complex and intimidating, to be honest. With all these smart people commenting on the issues, it’s hard to feel like you’re good enough to solve it!

Fast forward to the end of Mobile Makers, when I first learned iOS, and I made my very first open source contribution! Since then, I’ve made a few more, and I’m no longer scared of contributing to even more projects! Here is how I made the jump and what I learned in the process:

IT’S NOT MAGIC

Getting out of the Rails community, one thing I learned quickly was that there is actually a sort of mistrust of open source libraries. While I wanted to use all the CocoaPods all the time (just like in Rails!), the senior iOS engineer on my team would always question using external libraries, preferring to build our own if possible.

There are very few iOS libraries that are used consistently across iOS projects, and most projects only use a few libraries to start with! That’s changing a bit thanks to CocoaPods and how easy it is to delete/upgrade dependencies, but it’s nowhere as close to Rails level, where most of your project is made up of plug-and-play libraries.

Changing my mindset to realizing that these external libraries are not magic, has made a HUGE difference for me. Knowing that the person writing a library is a real person, who makes mistakes, might not write the best-optimized code, and who can’t see all the edge cases all the time makes it much easier to spot the small (or big) places where I can contribute.

CARING IS SHARING

While there are some people who advocate to actively look for open source projects to contribute to, I found that all the projects I contribute to are the ones I'm actually using in my own code.

The truth is, I don't have time to actively scour GitHub for issues/libraries to contribute to. But when I'm using a library in my own project, and I wish it had feature X or that bug Y was resolved, it's a no-brainer to solve it and contribute back! In fact, I get super excited when I find these opportunities, since I do like to contribute to open source!

Which brings me to my next point…

IT'S SUCH A GOOD FEELING…

Contributing to open source is seriously addictive! Already, knowing how to code and bending the computer to your will is magical and makes you feel like a sorceress. But when you can take someone else's "magical" code and make it better and they agree that you made it better (by merging it in), the feeling you get is indescribable. It's like you just became a Level 10 magician instead of Level 1.

START SMALL

Again, I thought of open source contributors as magicians making sweeping changes and improving everything with their magic wand, but the truth is most changes are pretty small. They just add up and eventually change and improve the whole library for everyone in a very substantial way. So don't underestimate the power of small changes!

Here are some recent examples of contributions I made/am in the process of making:

MODIFY A README

I wanted to possibly add the Toast library to my iOS project, but they didn't say on their README that there was a CocoaPod available. Since I just wanted to test out the library in my project, I wanted it to be easily removable. So even though I've used this particular library before and knew it was good, I started searching CocoaPods for another Toast library.

A few libraries down, I found that this particular Toast library was, in fact, a CocoaPod! To make sure other iOS developers know there is a CocoaPod available, I submitted a pull request with CocoaPod installation instructions to the README file of the library. Tiny change, but hopefully, it'll help other developers!

ADD ADDITIONAL SAME FUNCTIONALITY

When building an Android app for the CodePath final project, my team wanted to try out the current official Android font – Roboto. Turns out it's pretty manual to incorporate external fonts into Android, so we used a library called RobotoViews to help us out. Basically, each view has to be configured to have the Roboto typeface available.

However, there was one view that the RobotoViews library did not include that we needed – the newer Switch view. Adding the Switch view was just a lot of copy/paste following the convention of the other views, so it wasn't hard to add, but now another view is available as a RobotoView!

In other words, the creator of RobotoViews already did all the hard work to make it easy to modify only a few things to add a new view.

Similarly, I contributed to a popular iOS Foursquare client library by just adding an additional function that wasn't there originally, but was very easy to add based on the creator's work to abstract this process!

REFACTOR

I was adding a very small change to the ECSlidingViewController to make sure a keyboard is dismissed when a sliding menu slides out, when I noticed that three functions had the same exact code with a very slight variation. So I just refactored that code by creating a function those three functions can just call passing in the one different argument, so anyone who needs to make a change to this function in the future, only needs to make the change once.

As you can see, all my open source contributions are very tiny and easy to make! As you keep using external libraries, you'll see similar opportunities. So go ahead and make the tiny changes – they count!

HOW TO CONTRIBUTE

Watch this great step-by-step RailsCast on how to contribute to open source (it's a very similar process for non-rails projects). But basically, here are the steps:

FORK

Find the library you want to contribute to on Github, and simply click the Fork button! github-fork

CLONE

Next, clone the library that you forked – it should now be under your name (e.g. NatashaTheRobot/ECSlidingViewController), not the original creator! git-clone

BRANCH, CHANGE, PUSH

Once you've cloned the repository, change into the repository's folder. Next, check out a new branch with a good name reflecting the change you will make. Make the change, and push the branch to GitHub.

When you go to your GitHub profile main page, you'll see a big green "Compare and Pull Request" button. Take a look at your files, make sure everything looks good. Then make sure you make a pull request to the original branch (not the one you forked). When you’ve done the pull request, is should look something like this! Again, make sure you’re doing a pull to the original creator’s master branch! git-pull-request

TWEET

This step is of course optional, but I like to tweet at the creator to notify them of the change. It’s possible they don’t have GitHub notifications turned on, especially if it’s an older repository with less recent changes, so it’s nice to let them know and start a conversation.

They might also be too busy to merge it in, so it’s nice when they reply and tell you they’ll take a look when they have a chance. You know that your pull request will not be in limbo forever!

ENJOY!

Again, contributing to open source is really fun and a great way to learn from the best and improve your own skill. Hope this guide makes you feel less intimidated at contributing!

How to get started in civic hacking

Source: OpenSource.com

What is civic hacking?

Seventy people gathered together one sunny Oakland afternoon to volunteer and improve their city. There were no rakes or yard tools normally seen at volunteer-day events though. No paint brushes, no trash bags, no canned soup bins. These seventy people were laden with laptops and were volunteering to improve the city’s website.

This group of engaged citizens were building Oakland Answers, a new easy way to get answers for the most common questions asked on the Oakland City website. From finding out how to pay parking tickets, to checking what jobs the City is hiring for, the new website is citizen-focused and community built.

The day long event was called a "writeathon" and the majority of the folks in the room were not web developers but long time Oakland residents who came to write answers. Technologists were there too though, setting up servers and forking the open source code for the site. These web developers, answer writers, and City staff were all taking part in the growing new movement of civic hacking.

> Civic hacking is people working together quickly and creatively to help improve government.—Jake Levitas

Here are a few more examples of popular open source civic hacking projects:

- OneBusAway (on GitHub)
- City budget visualizations (on GitHub)
- 311 Service Tracker (on GitHub)
- Flu Shot Finder (on GitHub)

- StreetMix (on GitHub)
- CityVoice (on GitHub)

Open source civic hacking

Open source software is fundamental to civic hacking. Passionate volunteers write code and invent services that improve their own neighborhoods, but do so in a way that can be repeated in other communities around the world. Being able to easily share code without restriction is what allows for civic technology to scale.

For example, a few years ago in Boston there were severe snow storms that buried the fire hydrants. The same snow was downing powerlines and sparking fires. Some civic hackers saw this problem and created Adopt-a-hydrant (on GitHub), a way for neighbors to volunteer to shovel out the hydrants on their block. The following summer, the same code was forked and redeployed in Honolulu, not for snow but for tsunami sirens. Adopta has since been redeployed dozens of times and is being constantly improved by coders across the country.

Getting started

A great first project is to include your city in an existing service. Take Click that 'hood (on GitHub) for example. It’s a fun game that helps teach about a city’s neighborhoods. What's great about it is that it has clear instructions for adding your own city to the game. These instructions include using open source tools, collaborating on GitHub, and finding open data—all necessary skills for getting started in civic hacking.

Finding open data

The civic hacking movement is dependent on being able to easily find data about governments and the places they govern. If transit data, like bus schedules and train station locations, aren’t available than we couldn’t make any useful apps about transit. Luckily, many cities understand the importance of making their data available and have open data portals now. Data.gov has a list of many of the government data portals around the country and world. These portals gather all the available datasets that a city has and puts them all online in one place. The best data portals have that data in a machine readable format, so that it can be easily included in apps. Check your city’s website to find if a data portal exists. If not, then working with your city to get one setup is a great civic hacking project to start with.

Open source data portals:

- CKAN
- DKAN
- Socrata

Community

It’s important to remember that civic hacking includes both community and technology. All aspiring civic hackers need to join with others to solve our civic problems together. Check out the Code for America Brigade to find a local volunteer group or start your own. The best part of joining up with other civic hackers is finding out how they’ve achieved successes in their own cities. The Brigade is one of the best resources for discovering the latest open source tools and projects to work on. Also, working with city staff and civic leaders who are part of the Brigade is vital so that the civic technology apps created by civic hackers solve real societal problems. Many different skills and many different perspectives are needed to work on problems that effect many different people. Finally, to get really immersed in the civic hacking movement, consider applying for the Code for America Fellowship program.

Happy hacking!

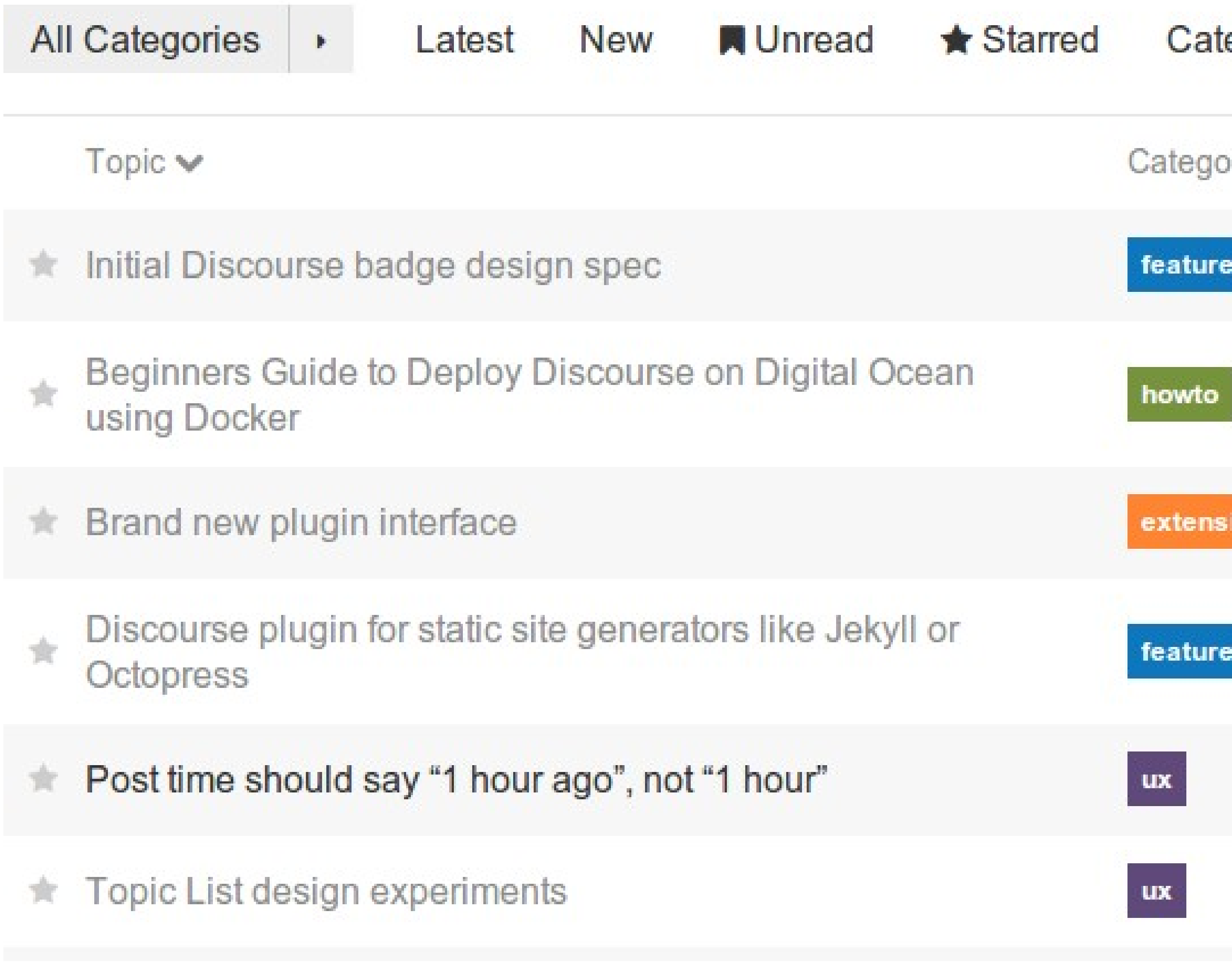
Getting started with Discourse

Source: Discourse wiki

This page covers basic usage of Discourse. Note that the interface may look different than the screenshots here, because your site admin may have customized the site.

Topic Lists

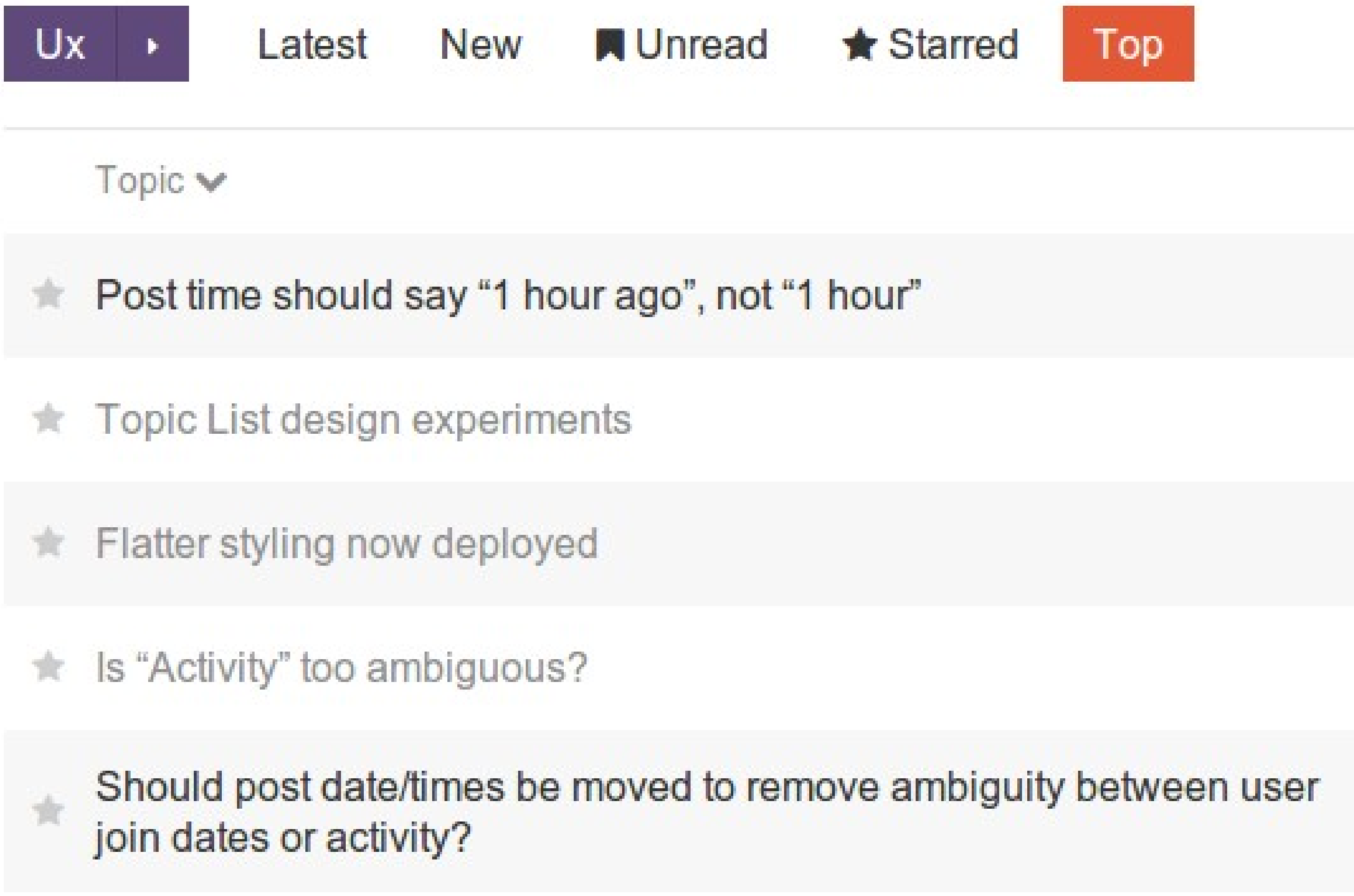
There are several topic lists in Discourse. The first one you see as a new user will be the *Top* list. It displays the 'best' posts from the past, by totaling up the topics' posts, likes, and views.



Topic	Catego
Initial Discourse badge design spec	feature
Beginners Guide to Deploy Discourse on Digital Ocean using Docker	howto
Brand new plugin interface	extens
Discourse plugin for static site generators like Jekyll or Octopress	feature
Post time should say “1 hour ago”, not “1 hour”	ux
Topic List design experiments	ux

There are several things on this page. On the first row, we have the **Category filter**, **Nav menu items**, and the **Create Topic** button.

You can use the **Category filter** to filter by category. For example, here is the top page, filtered to UX topics only:



Post time should say “1 hour ago”, not “1 hour”

Topic List design experiments

Flatter styling now deployed

Is “Activity” too ambiguous?

Should post date/times be moved to remove ambiguity between user join dates or activity?

A topic is displayed in a row on the list. Going from left to right, we have the **title**, the topic's **category**, a list of **participants**' avatars, the number of **posts** in the topic, the number of **likes** in the topic, the topic's **view** count, and its **creation** and **last activity** dates.

Clicking on the title or post count will bring you to the **last post that you read** in the topic. Clicking on the like count will take you to the **start of the topic in summary mode**, if available. Clicking the topic start date will take you to the **start of the topic**, and clicking the last activity date will take you to the **end of the topic**.

The other available topic lists are **Latest** (this will be the default once you achieve Trust Level 1), New, and Unread.

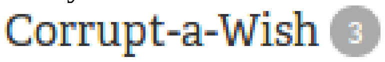
New and Unread topics

New topics are those that you have never read. By default, only topics created in the **last two days** will be considered New.

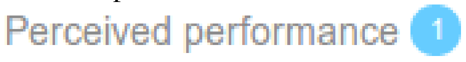
Topics considered New to you look like this: (screenshot)

There are two kinds of unread topics: those that you stopped reading, and those that have new posts.

Topics you stopped reading have a grey bubble with a number in it, representing the number of posts you haven't read yet.

It looks like this: 

Topics that have new posts since you last read them have a blue bubble with a number in it, which is the count of new posts.

It looks like this: 

Nav menu

The nav menu is displayed near the top of the topic lists:

The available tabs are Latest, New, Unread, Starred, Categories, Top, Read, and Posted. Not all of these will be shown, but they are still accessible.

- **Latest**: All topics, in order of last activity.
- **New**: Here, only topics considered new will be listed. (Logged-in only)
- **Unread**: Here, only topics considered unread will be listed. (Logged-in only)
- **Starred**: Only topics that you have starred will be listed. (Logged-in only)
- **Read**: Topics in the order that you last read them. (Logged-in only)
- **Categories**: Overview of the categories in the forum.
- **Top**: The most active topics in the last day, week, month, and year.

The leftmost item in the nav menu is the **default tab**, and it's what you see when you click the Discourse logo or a category badge. By default, this is the Latest tab.

Categories

Every topic has a **category**. Discourse, unlike older forums, encourages mostly flat categorization: there is only one subcategory level available, and all the topics go on the same list.

Categories page

The categories page gives you an overview of all the categories on the forum.

(screenshot)

Here we see that an entry in the categories page consists of the **category badge**, its **definition**, the badges of its **subcategories**, the avatars of repeat **participants**, a selection of recent **topics**, and the **rate of new posts and topics**.

Topic Page

Creating a topic

To create a topic, click on `+ Create Topic` in the upper-right of any topic list. If the topic list is filtered by a category, that category will be pre-filled for you.

Choose a good topic title and start writing in the composer.

Getting started with GitHub

Source: LifeHacker.com

> Dear Lifehacker, I've learned to code and want to start using GitHub to manage my projects. Despite the introductory lesson they provide, I still don't understand how it works at all. Can you help me?
>
> Sincerely, Git Help

Dear GH,

GitHub's a great tool but it's definitely a little confusing the first time around (and, possibly, a few times after that). That's likely why GitHub created software (for OS X and Windows) to make the process a bit easier. Nevertheless, it's good to learn the old-fashioned way otherwise your options in the simplified software won't make sense. Let's start by walking through the basics.

Step One: Sign Up for GitHub

Here comes the easy part: make yourself a GitHub account signing up on the front page. After completing the form, GitHub will sign you in and take you to your empty news feed. In the middle of the page, you'll see the boot camp (pictured to the right). We're going to go through it to set up your account and, later, create your first repository. Click on "Set Up Git" to get started.

Step Two: Install Git

GitHub exists because of a version control application called `git`. The site is based around how git works, and git is pretty old. It runs via the command line and has no fancy graphical user interface. Since it's made to manage code you wrote, this shouldn't sound too scary. (Of course, as previously mentioned, GitHub did make wonderful software to allow you to use their service without the command line but that won't help you too much unless you know the basics.)

Git works by reading a local code repository (just a folder containing code for your project) on your computer and the mirroring that code elsewhere (in this case, GitHub's servers). Initially we'll commit (i.e. send) your entire local repository to GitHub, but that's just a one-time affair. As you continue to work on your code, you'll simply commit changes. GitHub will then keep track of the changes you made, creating different versions of files so you can revert back to old ones if you want (or just keep track of those changes for other reasons). This is primarily why you'd want to use a version control system like git on your own, but additional benefits surface when using git to manage code with other people working on your project. When multiple developers commit code with git, GitHub becomes a central repository where all the code that everyone's working on can stay in sync. You'll commit your changes, and other developers will pull them (i.e. sync them to their local repository). You'll do the same with their code.

Git makes this all happen, so you need to download the latest version and install it. On OS X, you'll just install the command line app. On Windows, you'll get a few more items. We'll discuss how they work in the next step.

Step Three: Set Up Git

To set up git, you need to make your way into the command line. On OS X, that means launching the Terminal app (Hard Drive -> Applications -> Utilities -> Terminal) and on Windows that means launching the Git Bash app you just installed—not the Windows command prompt. When you're ready, tell git your name like this:

```
git config --global user.name "Your Name Here"
```

For example, mine would look like this because I'm using a test account for this example:

```
git config --global user.name "Adam Dachis"
```

You can put in any name you like, but afterwards you'll need to input your email and that email must be the email you used when signing up for GitHub:

```
git config --global user.email "your_email@youremail.com"
```

If, for whatever reason, you signed up for GitHub with the wrong email address, you'll need to change it.

Now, to avoid always entering your login credentials and generating SSH keys, you'll want to install the credential helper so your passwords are cached. If you're on Windows, download it and install it. If you're on OS X, you'll need to handle this through the Terminal. To start, use this command to download the credential helper:

```
curl -s -O \
http://github-media-downloads.s3.amazonaws.com/osx/git-credential-osxkeychain
```

This will download a tiny little file and shouldn't take too long. When finished, enter the following command to make sure the permissions are correct on the file you just download (and fix them if not):

```
chmod u+x git-credential-osxkeychain
```

Now it's time to install the credential helper into the same folder where you install git. To do so, enter this command:

```
sudo mv git-credential-osxkeychain `dirname \`which git\``
```

You'll be prompted for your administrator password because the above command began with sudo. Sudo is shorthand for "super user do" and is necessary when performing a task that requires root access. The sudo command allows you to become the root user (a user with permission to do pretty much anything) on your operating system for a moment so you can perform this task. You're asked to enter your password to prove you're an administrator on the computer and should be allowed to do this. Once you've entered your password and the credential helper has been moved, finish up the installation with this command:

```
git config --global credential.helper osxkeychain
```

Now you're all set and can move on to actually using git and GitHub!

Step Four: Create Your First Repository

Now that you've made it this far, you can actually use GitHub! As a first order of business, we're going to create a repository (or "repo" for short). Head on over to GitHub and click the "New Repository" button on the top right of your account page. (Note: If you're still displaying the

GitHub bootcamp section, it'll show up underneath it.)

When creating a repository you have a few things to decide including it's name and whether it'll be publicly accessible or not. Choosing a name should be pretty simple because you likely already have a name for your project. If you're just following along for learning purposes, use "Hello-World." Why "Hello-World" and not "Hello World"? Because spaces and special characters will cause problems. Keep it simple and easy to type in the command line. If you want to include a more complex name, you can add it to the optional description field beneath the name field.

If you're creating an open-source project, you want a public repository. If you want to code by yourself or share only with specific people, a private repository will do. Make the choice that works best for you and your project.

When you're all done, you can click the "Create repository" button but you might want to do one other thing first: check the "Initialize this repository with a README" checkbox. Why? All repositories require a README file. Ideally that file would contain a little information about your project, but you might not want to deal with that right now. By initializing the repository with a README, you'll get an empty README file that you can just deal with later. For the purposes of this tutorial, we're going to leave the box unchecked because, in the next section, we're going to create a README file from scratch to practice committing (sending) it to GitHub.

Step Five: Make Your First Commit

When you send files to GitHub, you *commit* them. To practice, we're going to initialize your local repository and create a README file to commit as practice. Before you start, you need to know where your local code repository is on your computer and how to access it via the command line. In this tutorial, we're going to assume there's a directory called "Hello-World" in your computer's home folder. If you need to create one, just run this command (same for Git Bash on Windows and OS X's terminal):

```
mkdir ~/Hello-World
```

Now change to that directory using the cd (change directory) command:

```
cd ~/Hello-World
```

In case you were wondering, the ~ represents your home directory in Git Bash and Terminal. It's simply shorthand so you don't have to type it all out (which would look more like /Users/yourusername/). Now that your repository is ready, type this:

```
git init
```

If you already had a repository ready to go, you'd just need to cd to that directory and then run the `git init` command in there instead. Either way, your local repository is ready to go and you can start committing code. But wait, you don't have anything to commit! Run this command to create a README file:

```
touch README
```

Let's take a break for a second and see what just happened. Go into the home folder on your computer and look at the Hello-World folder (or look at whatever folder you're using for a local repository). You'll notice a README file inside, thanks to the command you just ran. What you won't see is a .git folder, but that's because it's invisible. Git hides it in there, but because you ran the `git init` command you know it exists. If you're skeptical, just run the `ls` command in Git Bash/Terminal to display a list of everything in the current directory (which, if you're following along, is your local repository).

So how does git know we want to commit this README file we just created? It doesn't, and you have to tell it. This command will do the trick:

```
git add README
```

If you want to add other files to commit, you'll use the same command but replace README with the name of a different file. Now, run this command to commit it:

```
git commit -m 'first commit'
```

While the other commands were pretty straightforward, the commit command has a little more going on so let's break it down. When you type `git`, that's just telling the command line that you want to use the git program. When you type `commit`, you're telling git you want to use the commit command. Everything that follows those two thing count as options. The first, -m, is what's known as a flag. A flag specifies that you want to do something special rather than just run the commit command. In this case, the -m flag means "message" and what follows it is your commit message (in the example, 'first commit'). The message isn't absolutely necessary (although you'll usually need to provide one), but simply a reference to help you differentiate the various versions of a file (or files) you commit to your repository.

Your first commit should go by in a split second because you haven't actually uploaded anything yet. To get this empty README file to GitHub, you need to push it with a couple of commands. Here's the first:

```
git remote add origin https://github.com/yourusername/Hello-World.git
```

You need to replace "yourusername" with—you guessed it—your GitHub username. For me, it'd look like this:

```
git remote add origin https://github.com/gittest1040/Hello-World.git
```

This command tells git where to send your Hello-World repository. Now all you need to do is send it:

```
git push origin master
```

Once you run that command, everything (in this case, just your README file) will make it's way over to GitHub. Congratulations on your first commit!

Learning More

Using GitHub requires more than just committing a README file, but these basics should give you a good grasp on how to interact with the git app and the service. Now that you know how GitHub works at its core, you can use the GitHub apps to manage your code instead if you prefer. If you want to learn more about GitHub, there are some great tutorials. For starters, take a look at how to fork a repository and LockerGnome's GitHub guide.

Have fun managing your code!

Love, Lifehacker

# Additional resources from GitHub
Good Resource for learning git and github link # Going open source on Android with F-Droid
Source OpenSource.com

Android. It can be a divisive word in the free and open source software world. Some embrace it, others shun it. Some still use open versions of Android like Cyanogenmod and Replicant. If you do use an Android device—no matter what version of the operating system it is—there's one thing that you need to get the most out of your device: apps. There's just no way around that.
Most people grab their Android apps from the Google Play Store. Some might install apps from the Amazon Appstore or another third-party source. A majority of the apps that you get from Google and Amazon's app stores are proprietary, and many of them collect a lot of information about you.
So what choice to you have if you want to use Android and keep your apps as free and open as possible? You turn to F-Droid.
## F-Droid?
F-Droid is:
>an installable catalogue of FOSS (Free and Open Source Software) applications for the Android platform.

All of the apps are FOSS and only FOSS. The source code is available, with app listings often pointing you to where you can download it. F-Droid also warns you if an app uses or relies upon a proprietary service.
F-Droid Non-Free Warning
## Getting going
You can do this in two ways: Download and install the F-Droid client, or download the .apk installer for the app that you want and install it by hand. In either case, you might need to enable the installation of third-party apps on your device. To do that, tap settings. Then, tap security. Finally, select the unknown sources option.
I prefer to use the F-Droid client because it makes searching for and updating apps a lot easier.
## Using F-Droid
Let's assume that you plan to use the F-Droid client to install your apps. Once you've installed the client, fire it up. It may take a few moments (or longer) for the database of apps to refresh and to check whether or not the apps in the catalogue are compatible with your device.
Once that's done, you're ready to go.
## Apps, apps, apps
The first thing you'll notice is the number of apps available with F-Droid: just over 1,300, in contrast to the hundreds of thousands that are available in the Google Play Store. You won't find many of the popular apps that you may have grown to know and love in the F-Droid repository, but that doesn't mean you won't recognize some. These include Firefox, ownCloud, VLC Media Player, DuckDuckGo, K-9 Mail, and FBReader.
The apps are divided into a dozen categories, ranging from education to games, to internet, to office and productivity apps. Tap the right-facing triangle in the top corner of the F-Droid window to open a list of the categories.
F-Droid Categories
From there, tap a category and then tap the app that you want to install. F-Droid downloads the installer. You'll be shown a list of the permissions the app needs (if any) and whether or not you want to proceed. When you're ready, tap **Install**.
## Maintenance and such
The main window of the F-Droid app has tabs that list the available apps, the apps you've

installed, and the ones you've installed that have updates available. You can remove unwanted apps and update installed apps, with just a tap (or two). You'll also want to keep the list of available apps up to date. To do that, tap the **Refresh** icon to load an updated listing.

So why would you want to do this, besides a desire to be on the bleeding edge? While new apps aren't added to F-Droid all that regularly, the number of apps available through F-Droid jumped from about 1,200 to 1,340 in the past month and a half or so.

F-Droid may not have the breadth of apps available in the Google Play Store and other third-party Android software libraries, but if you want to use as many free and open source apps with your Android device as you can, then it's an option you'll want to explore.

# How to get started in Open Source (in 2014)

Source: OpenSource.com

For open source projects to survive and thrive, it takes all sorts of different people contributing in various ways. We hope you are already participating in the open source community, but if you’re not, 2014 is going to be a great year to start!

Already an open source convert? Why not consider giving back in a new way this year?

Here are seven roles you might consider taking on in the new year.

## 7 ways to get involved in open source in 2014

1. **Get your start as an open source user**. First and foremost, we hope you're an open source user! There are open source solutions for nearly every need you can imagine, and chances are you've already used a few. From browsers like Firefox to email clients like Thunderbird, and content management systems like Drupal and Wordpress, much of the software that powers the Internet is open source. Why not give a new tool a try this year?
1. **Share your favorite project with the world as an open source promoter**. The open source community and every project within it needs cheerleaders to spread the word about the benefits of the open source way. Are you a social media junkie? Join us on Facebook, Twitter, or your favorite social media platform, and share your own open source story with the world!
1. **Put your digital quill to work as an open source writer**. Writing for open source projects can take many different forms, but two roles that are almost always in need are the documentation writer and the translator. There are lots of tools available to help you document open source projects. A couple we've profiled this year are: Zanata for translators and AsciiDoc for documentation.

For more general-purpose writing tools, LibreOffice is a complete office suite that can do word processing and so much more!

1. **Let your artistic skill shine as an open source designer**. Do you have a knack for design? A strong interface and visual cues can make or break an open source project, and many projects could benefit from your help. If you've got the skills but not the tools, check out GIMP (the GNU Image Manipulation Program) or Inkscape to get you started. Both run on several different platforms, including Windows, Mac OS X, and Linux.
1. **Keep software bug-free as an open source tester**. You don't have to be a quality assurance expert to get started with software testing. If you're using open source software already, you're already halfway there to being a tester. All it takes is sharing your experience if something doesn't seem quite right. Most projects have bug tracking applications that are easy enough for even the novice user to report an issue or make a feature request.

Thrilled to see a new feature added to your favorite program? Don't forget to say thank you.

1. **Help others learn the ropes as an open source community leader**. You can also help build the open source movement by participating in a project's community. Many projects have their own dedicated forums or mailing lists, and there are plenty of spots on the net for participating more generally. Share you experience with others by helping answer questions from all sorts of users, from novice to expert. Stack Exchange hosts conversations around a variety of open source tools.
1. **Write code as an open source developer**. We say it a lot around here that the open source way applies to much more than just software development. But code still matters! If you haven't tried your hand at programming yet, 2014 should be your year to get started. There are tons of great resources out there for beginners to get started and for fledgling programmers to further build their skillsets.
Go here to learn to program the open source way?
### Take your pick
With so many different options for participating, anyone can join the open source community. How are you going to make 2014 your year of open source? Pick your role, and feel free to pick more than one! Need more guidance and inspiration? Opensource.com has these resources to help you get started.
- How non-programmers can contribute to open source projects - 10 ways to contribute to an open source project without writing code - 10 ways to start contributing to open source - 10 ways to get started with open source # Key books to read
Source: Open Source Books
### Open Source Essentials Many authors have chronicled the history of open source and profiled the movement’s most prolific personalities. These are some of the most popular accounts. If we taught a class—call it "Open Source 101"—then this list would be the syllabus.
- The Cathedral and the Bazaar by Eric S. Raymond - Free Software, Free Society by Richard M. Stallman - Just for Fun by Linus Torvalds and David Diamond - Rebel Code by Glyn Moody - Producing Open Source Software by Karl Fogel - Free for All by Peter Wayner - Free as in Freedom by Sam Williams - Open Sources by Chris DiBona and Sam Ockman - The Foundation for an Open Source City
### An open world: Stories from the open source community [source link] At opensource.com, our mission is to shine a light on the places where the open source way is magnifying ideas and multiplying effort. And we remain especially interested in topics beyond technology—developments in areas like business, education, government, health, law, and everyday life. This collection offers some of our most empowering stories. From the trenches. From the library. From the cubicle. From the capitol. From the classroom and the boardroom and the courtroom. These are our community members—and perhaps you could be in the next edition. Download this free eBook An open world: Stories from the open source community [ePub] | [PDF]
### How to get started with Open Source [source link] Taking the first steps in any journey can be scary. There are new obstacles to hurdle, unfamiliar landscapes to traverse, and usually the destination is shrouded in mystery. But that's also what makes new ventures exciting and worthwhile.
The collection of stories in this eBook are about striking out on the open source way. Whether you're an individual, seeking information on moving away from closed software, or an

organization looking for free and open alternatives to the utilities that help your business succeed, these are stories about finding your footing in the world of open source. Several are “origin stories” about how experts discovered open source, and how they've flourished in the time since. Others are focused guides on finding the right software, using a specific open source tool, or introducing your neighborhood or town to the power of the open source way.
Remember that it's not easy to get started with anything new, but, as an old Buddhist saying goes, “It is better to travel well than to arrive.” We hope these stories will help you travel well along the open source way, and keep you traveling for years to come.
Download this free eBook How to get started with open source [ePub] | [ODT]
### Open always wins: A Michael Tiemann collection [source Link] Michael Tiemann is Vice President of Open Source Affairs at Red Hat and former President of the Open Source Initiative. He has written many articles for Opensource.com speaking to the power of the open source way, and part of his process has been to highlight the great ideas of others.
From his introduction:
Linus Pauling famously said “The best way to have a great idea is to have lots of ideas.” This is easier said than done, for many reasons. For me, the foremost reason is that nobody wants to be known for having a dumb idea, so we self-edit. If we self-edit too much, we end up having not a lot of ideas, so having a great idea becomes nearly impossible.
A second challenge is creating a space where ideas can combine, coalesce, catalyze, evolve, and, if they are truly great ideas, crystalize. Is that best space an isolated office where the mind of a lone genius can evaluate all and choose correctly the One Best Idea? Or is it better to open up the process to a diverse set of perspectives, up to and including every possible stakeholder in the outcome?
Glean insight from a true open source visionary and learn how can you apply those principles to your business, career, life, hobbies, and more with this eBook.
Download this free eBook Open always wins [ePub] | [ODT]
### Applying open source principles to government [source link] At opensource.com, we look at the intersection of open source and government, with a special focus on the ways governments adopt and release new technologies and can cultivate open source communities. We document the way that open source principles—participation, transparency, collaboration, sharing, meritocracy, community, and rapid prototyping—have enormous value beyond the technology sector. Citizen movements around open voting, the Freedom of Information Act (FOIA) request tracking, and even crowdsourced legislation, represent a growing trend of open source principles within government.
Download the free eBookThis collection features stories about government initiatives around the world, from the first two years of opensource.com. They illustrate the impact of open source on government and vice versa. There is a long journey left and we hope you'll share your experiences with us.
Download this free eBook Applying open source principles to government [ePub] | [PDF]
### Opensource.com favorite reads Each year, the OpenSource.com community collaborates on an open source summer reading list. Each one is filled with recommendations for books our readers, advocates, and experts feel exemplify the open source way. Browse the lists to find your new favorite read.
- The 2010 annual reading list - The 2011 annual reading list - The 2012 annual reading list -

The 2013 annual reading list - The 2014 annual reading list # Open-Source Foundations
## About this section This collection of articles is designed to introduce you to general concepts within open source and civic hacking.
## Articles - What is Open Source - Beginners Guide to Open Source - Getting started in Civic Hacking - How to get started in Open Source - Three key elements that define every open source project - Getting started with GitHub - Getting started with Discourse - Going open source on Android with F-Droid - Things newcomers to open source rarely ask but often wonder - Key Books to Read and Open Source Organizations # Things newcomers to open source rarely ask but often wonder Source: OpenSource.com

Open Source Comes to Campus is an event series run by OpenHatch that introduces college students to open source tools, projects, and culture. Whenever we get a new-to-us question at an event, we write it down and answer it more fully on our blog. Here's a collection of "Infrequently Asked Questions" that are especially relevant for newcomers to open source projects.

Q: I'm worried that I'll be a burden on a project because I'm so new. What kind of effort does a project have to make to build an open source community?

A: First of all, you should know that expert open source community builders like Linus Torvalds understand this tension between training new contributors and the time it takes from existing maintainers to mentor them. I'll quote two paragraphs from a 2004 post Linus made on the Linux kernel list:

> On Tue, 21 Dec 2004, Jesper Juhl wrote:

> Should I just stop attempting to make these trivial cleanups/fixes/whatever patches? are they more noise than gain? am I being a pain to more skilled people on lkml or can you all live with my, sometimes quite ignorant, patches? I do try to learn from the feedback I get, and I like to think that my patches are gradually getting a bit better, but if I'm more of a bother than a help I might as well stop.

> Linus replied:

> To me, the biggest thing with small patches is not necessarily the patch itself. I think that much more important than the patch is the fact that people get used to the notion that they can change the kernel—not just on an intellectual level ("I understand that the GPL means that I have the right to change my kernel"), but on a more practical level ("Hey, I did that small change").

Another thing to keep in mind that is, for most projects, the project leader is the only contributor (source). Consider this: The act of bugging them with questions as a newcomer can make them more enthusiastic about the project as a whole!

If it were trivial to build an easily accessible open source community, virtually all projects would do it. There is some effort involved. Of course, a lot of the work that makes a project welcoming to newcomers—good documentation and setup guides, a well-maintained issue tracker, active development, standards of conduct in the community—makes the project better for everybody. But it does take time and energy, which many communities of projects aren't willing to expend. That's why OpenHatch exists! We know that some projects put more effort into welcoming contributors, and we want to help you find those projects.

The kinds of projects that welcome newcomers don't see you as a burden. They see you as a huge help—even when you're struggling to understand bugs in the issue tracker or get the development environment set up. When you ask a member of an open source project for help, that gives them important information about what part of their project is confusing or incorrect. The questions you ask and guidance you need lets them know how help others later on. And of course, once you've gotten familiar with the project, you'll be able to help even more. The right kinds of project see that potential in you and want to work with you to get there.

Two rules of thumb for people who worry about being too much of a bother (which includes us, sometimes!): First, if people on the mailing list or the bug tracker haven't told you to stop talking, then you're probably okay, even if they haven't replied to you yet. Second, if you want another opinion, just join the #openhatch IRC channel and ask us.

Q: How do open source projects review changes, and how do those changes differ from a process like that of Wikipedia?

A: One key cultural element of Wikipedia, at least so far in its history, is that anyone's edits to English Wikipedia immediately become part of the live version of the encyclopedia. By contrast, to maintain quality, open source projects typically permit only a handful of people to *directly merge* changes into the main project. Therefore, submitted changes go through review. Different communities have different standards and processes. Sometimes, there is no review at all, just automatic merging by the maintainer. In other cases, like Linux, submissions go to a mailing list for review. Others, like the OpenHatch web app, use web-based tools like GitHub pull requests. A great reference for a more complicated process is the OpenStack Gerrit Workflow documentation page.

Q: How do you make time to contribute to open source when you're a busy student?

A: By simply using open source programs, you are uniquely positioned to help other users. Don't forget that helping people use the software is a substantial contribution to the community! By becoming an expert in whichever programs you use, you will also gain the knowledge to help convert bugs filed by others into actionable reports for the main developers. And once you reach that point, you might find it easier to just fix the issue!

That said, it can be daunting to add another activity to schedules already stuffed full of classes, paid work, and student life. One way to approach open source is to see it not as an alternative to doing schoolwork but as part of your education. For computer science majors, open source projects are a great way to practice the concepts you're being taught in class. In the others sciences, learning how to use and contribute to open source tools such as R or Octave (or ImageJ or PsychoPy or JMARS) can help you excel when doing laboratory work—and it looks great on your resume. Even in the arts and humanities, learning about open source tools such as Processing can help you succeed.

For students who work while they're at school, open source projects can be potential employers. You can apply for Google Summer of Code or the GNOME Outreach Program and do a paid internship at an open source project. Individual companies working with open source may hire students that show enough interest and ability. You can even employ yourself using open source —for instance, you could do freelance work making websites and applications by using open source tools like WordPress and Drupal.

You can make open source a part of your social life as well. Join (or start) a Students For Free Culture chapter on your campus or invite friends over for a 'bug-fixing party'.

Finally, remember that your contributions to open source can be as big or as small as you want them to be. If you only have time to spend a few hours a month writing documentation or fixing small bugs in a project, that’s great—the skills you’re learning and connections you’re making will serve you well if you ever decide to get more involved.

Q: What is the difference between source downloads for developers and downloads for users? What’s a stable release?

A: Open source projects often release the program in a few different ways. Typically, if you are trying to contribute, you will want the latest source code cloned or 'checked out' from the project’s version control system. If you are just trying to use the program, you typically want a user-oriented download.

Many programs’ source code can’t be executed directly on a computer—it must be 'compiled', which turns it from human-readable text into machine-executable binary. The most common languages where this is true are Java, C, and C++. Therefore, the most useful download for a person who wants to run a project like OpenOffice, which is mostly written in C++, would be a compiled version specific to their operating system (for example, a Windows-specific build). By contrast, if you want to contribute to the program, you should act similarly to how the main developer acts—that is, by using a version control system on the editable program text, the source code.

Another important concept is the stable release. During development, people make contributions that break the program, conflict with other changes, or are simply unfinished. At any given point the most recent version might be completely unusable. So, maintainers periodically work towards releases (noun), which are tested to make sure they’re usable. If they are, maintainers release (verb) it.

So if you’re a user, you’ll want the stable release most appropriate to your operating system and computer. If you want to contribute, you’ll want the latest, non-compiled version.

Q: Are all open source projects welcoming to newcomers? How can you tell when the communities around a project are filled with jerks?

A: Not all open source projects are welcoming to newcomers. Not all open source projects need to be. Some projects benefit from keeping their communities small and experienced. And, some maintainers simply don’t have the energy or the inclination to mentor new people.

There’s nothing wrong with that, but it’s a shame when newcomers try to get involved with a project that isn’t interested in involving them. It’s a frustrating experience for everyone and unnecessary when there are plenty of people who’d be excited to work with them.

So how can you find a good project for you, a newcomer, to contribute to? Here are some good signs to look for:

- A large, active community is more likely to have members who can take the time to mentor. It also means that contributions will be acted on more quickly. - Good documentation and demos mean that developers have thought about how to introduce people to their project. - Some projects have Codes of Conduct/Diversity Statements, which demonstrate that the community is trying to be a safe and welcoming space. - Projects that are part of Google Summer of Code or the GNOME Outreach Program for Women have made a commitment to being good environments for newcomers.

At OpenHatch, we try to identify projects that are especially good for newcomers. We’re also developing a list of OpenHatch-affiliated projects that are working with us to make contributing to their projects as easy as possible. (If you’d like to be on that list, let us know!) Feel free to contact us and ask us for recommendations or if a particular project is known to be good for

newcomers.
Of course, there’s a difference between not having the time and energy to help newcomers and being actively mean or even abusive about it. If you experience the latter and our comfortable telling us about it, we’ll do our best to support you. OpenHatch people are always willing to provide advice on projects to join, either to help you find a great experience or to make the most of a bad experience. Find us on on IRC (#openhatch on irc.freenode.net) or email us at hello@openhatch.org. You can also join groups such as Systers or the Empowermentors Collective, which may help you navigate the free software community.
### More resources
This article has been a collection of information gathered during Open Source Comes to Campus events:
- Infrequently Asked Questions: Wellesley College - Infrequently Asked Questions: UMass Amherst - Infrequently Asked Questions: Bloomington
Read more posts about this event series on our blog.
For blog posts about getting involved with OpenHatch, read about the experiences of two of our contributors: Britta and Mandar. # Three key elements that define every open source project
Source: OpenSource.com
Open source has come a long way in the past 30 years and is entering the consciousness of most modern cultures. When thinking of open source projects, people categorize them several ways: governance structure, type of product platform, programming language, utility, technical details (language written in), industry sponsored or fully independent, and more.
But what truly defines any open source project, making it a unique entity different from all other open source projects? I would propose that there are three key elements of any open source project that frame, define, and differentiate that project from all others: the code, the community, and the brand.
## The code
Code is king. Code is what makes a product do something, and that's why open source projects exist in the first place: to build something useful. Technologists get excited about what the code does, and how it does what it does. Marketers get excited about how the product will solve their customers' problems. Code is what most people are looking for when they're searching for an open source project to use.
Sounds simple enough—so why don't we define an open source project purely based on its code? As anyone who has worked in software development already knows, code is ever-changing and ephemeral. In the open source frontier, free from the traditional controls of corporate-led projects, "the code" can get very hard to follow: open source code is infinitely forkable. Once your code is checked in under an Open Source Initiative (OSI) license to a public repository, it is fully accessible to anyone and everyone to take—and modify—for their own purposes. Once another user forks the code from your project and makes a slight modification, it is no longer officially part of your original project.
## The community
If code is the "what" of a project, then the community is the "who" of the project—the people who make it all happen. The core community of a project includes anyone actively involved in moving the project forward, such as the engineers writing the code and the end users who provide feedback or request specific modifications. The overall community also includes people

who don't check code, but provide support such as governance/process oversight, public relations/marketing, training, or financial or employment support. The social norms, the etiquette, and the ethos of the community help to differentiate a project from all others.
While participation in an open source project may be part of some individuals' paid employment (e.g. a corporate-employed software engineer assigned to work on an open source project for a certain percentage of her or his time), most open source community members participate voluntarily with no direct connection to their paycheck. So members tend to come and go as their interest or their other commitments wax and wane, or as their employer changes strategy. Like the code, the community is ever-changing.
Unlike a corporate software development project that can plan on having employees with certain skills available to assign to do specific work, the participation in an open source community is unpredictable and often beyond the control of the project. Personality conflicts arise and may result in highly skilled contributors leaving the community more readily than they would leave a paid employment. But the benefits of an open community can be seen in the enthusiasm and drive of many community members, and in the longevity of successful project communities, and in syncrhonicity and great work forward on the code.

## The brand

Brand is how the world outside of an open source project learns about that project. When individuals or companies are deciding which project to use or to invest in, branding helps them differentiate projects that offer similar functionality. Of course they will consider other details, but it is much easier to think, "Do I want to support Hadoop, with the yellow elephant?" rather than "Do I want to support Cloudera's CDH or the Hortonworks Data Platform, or the newly announced ODP?"
"The brand" includes many things: the official name of the project, a logo for the project or product, and even the appearance of the project's website and your product's UI. Some branding components in particular are legal trademarks: these typically include the official software product name and logo, although trademarks are strongest when used consistently.
Unlike code and community, a project's brand is not ever-changing or ephemeral. A trademark cannot be forked without legal permission, and a project brand can stay consistent even when community membership fluctuates. In many ways, the brand and trademarks are the element of a project that can be most easily controlled and maintained. However, proper trademark usage can be overlooked—or under appreciated by the community inside of the project—as an important tool for defining a project's unique character. Given that anyone can fork the code, and that community members come and go, a project's brand and trademarks are crucial elements for maintaining a project's longevity and independence, and to continue to draw in new project contributors.

# What is Open-Source? Source: OpenSource.com

The term "open source" refers to something that can be modified because its design is publicly accessible.
While it originated in the context of computer software development, today the term "open source" designates a set of values—what we call *the open source way*. Open source projects, products, or initiatives are those that embrace and celebrate open exchange, collaborative participation, rapid prototyping, transparency, meritocracy, and community development.

## What is open source software?

Open source software is software whose source code is available for modification or enhancement by anyone.
"Source code" is the part of software that most computer users don't ever see; it's the code computer programmers can manipulate to change how a piece of software—a "program" or "application"—works. Programmers who have access to a computer program's source code can improve that program by adding features to it or fixing parts that don't always work correctly.

## What's the difference between open source software and other types of software?

Some software has source code that cannot be modified by anyone but the person, team, or organization who created it and maintains exclusive control over it. This kind of software is frequently called "proprietary software" or "closed source" software, because its source code is the property of its original authors, who are the only ones legally allowed to copy or modify it. Microsoft Word and Adobe Photoshop are examples of proprietary software. In order to use proprietary software, computer users must agree (usually by signing a license displayed the first time they run this software) that they will not do anything with the software that the software's authors have not expressly permitted.
Open source software is different. Its authors make its source code available to others who would like to view that code, copy it, learn from it, alter it, or share it. LibreOffice and the GNU Image Manipulation Program are examples of open source software. As they do with proprietary software, users must accept the terms of a license when they use open source software—but the legal terms of open source licenses differ dramatically from those of proprietary licenses. Open source software licenses promote collaboration and sharing because they allow other people to make modifications to source code and incorporate those changes into their own projects. Some open source licenses ensure that anyone who alters and then shares a program with others must also share that program's source code without charging a licensing fee for it. In other words, computer programmers can access, view, and modify open source software whenever they like—as long as they let others do the same when they share their work. In fact, they could be violating the terms of some open source licenses if they don't do this.
So as the Open Source Initiative explains, "open source doesn't just mean access to the source code." It means that anyone should be able to modify the source code to suit his or her needs, and that no one should prevent others from doing the same. The Initiative's definition of "open source" contains several other important provisions.

## Is open source software only important to computer programmers?

Open source software benefits programmers and non-programmers alike. In fact, because much of the Internet itself is built on many open source technologies — like the Linux operating system and the Apache Web server application — anyone using the Internet benefits from open source software. Every time computer users view webpages, check email, chat with friends, stream music online, or play multiplayer video games, their computers, mobile phones, or gaming consoles connect to a global network of computers that routes and transmits their data to the "local" devices they have in front of them.
The computers that do all this important work are typically located in faraway places that users don't see or can't physically access—which is why some people call these computers "remote computers." More and more, people rely on remote computers when doing things they might otherwise do on their local devices. For example, they use online word processing, email

management, and image editing software that they don't install and run on their personal computers. Instead, they simply access these programs on remote computers by using a Web browser or mobile phone application.
Some people call remote computing "cloud computing," because it involves activities (like storing files, sharing photos, or watching videos) that incorporate not only local devices, but also the global network of remote computers that form an "atmosphere" around them. Cloud computing is an increasingly important aspect of everyday life with Internet-connected devices. Some cloud computing applications, like Google Docs, are closed source programs. Others, like Etherpad, are open source programs.
Cloud computing applications run "on top" of additional software that helps them operate smoothly and effectively. The software that runs "underneath" cloud computing applications acts as a *platform* for those applications. Cloud computing platforms can be open source or closed source. OpenStack is an example of an open source cloud computing platform.
## Why do people prefer using open source software?
Many people prefer open source software because they have more control over that kind of software. They can examine the code to make sure it's not doing anything they don't want it to do, and they can change parts of it they don't like. Users who aren't programmers also benefit from open source software, because they can use this software for any purpose they wish—not merely the way someone else thinks they should.
Others like open source software because it helps them become better programmers. Because open source code is publicly accessible, students can learn to make better software by studying what others have written. They can also share their work with others, inviting comment and critique.
Some people prefer open source software because they consider it more secure and stable than proprietary software. Because anyone can view and modify open source software, someone might spot and correct errors or omissions that a program's original authors might have missed. And because so many programmers can work on a piece of open source software without asking for permission from original authors, open source software is generally fixed, updated, and upgraded quickly.
Many users prefer open source software to proprietary software for important, long-term projects. Because the source code for open source software is distributed publicly, users that rely on software for critical tasks can be sure their tools won't disappear or fall into disrepair if their original creators stop working on them.
## Doesn't "open source" just mean something is free of charge?
No. This is a common misconception about what "open source" implies. Programmers can charge money for the open source software they create or to which they contribute. But because most open source licenses require them to release their source code when they sell software to others, many open source software programmers find that charging users money for software services and support (rather than for the software itself) is more lucrative. This way, their software remains free of charge and they make money helping others install, use, and troubleshoot it.
## What is open source "beyond software"?
At opensource.com, we like to say that we're interested in the ways open source can be applied to the world beyond software. We like to think of open source as not only a way to develop and

license computer software, but also an attitude. Approaching all aspects of life "the open source way" means expressing a willingness to share, collaborating with others in ways that are transparent (so that others can watch and join too), embracing failure as a means of improving, and expecting—even encouraging—everyone else to do the same.
It means committing to playing an active role in improving the world, which is possible only when everyone has access to the way that world is designed. The world is full of "source code"—blueprints, recipes, rules—that guide and shape the way we think and act in it. We believe this underlying code (whatever its form) should be open, accessible, and shared—so many people can have a hand in altering it for the better.
Here, we tell stories about what happens when open source values are applied to business, education, government, health, law, and any other area of life. We're a community committed to telling others how the open source way is the best way—because a love of open source is just like anything else: it's better when it's shared.
## Where can I learn more about open source?
We recommend visiting the Opensource.com resources page.
http://opensource.com/resources/what-open-source
# Android Design Principles Source: Android.com
These design principles were developed by and for the Android User Experience Team to keep users' best interests in mind. For Android developers and designers, they continue to underlie the more detailed design guidelines for different types of devices.
Consider these principles as you apply your own creativity and design thinking. Deviate with purpose.
## Enchant Me ### Delight me in surprising ways A beautiful surface, a carefully-placed animation, or a well-timed sound effect is a joy to experience. Subtle effects contribute to a feeling of effortlessness and a sense that a powerful force is at hand.
### Real objects are more fun than buttons and menus Allow people to directly touch and manipulate objects in your app. It reduces the cognitive effort needed to perform a task while making it more emotionally satisfying.
### Let me make it mine People love to add personal touches because it helps them feel at home and in control. Provide sensible, beautiful defaults, but also consider fun, optional customizations that don't hinder primary tasks.
### Get to know me Learn peoples' preferences over time. Rather than asking them to make the same choices over and over, place previous choices within easy reach.
## Simplify My Life ### Keep it brief Use short phrases with simple words. People are likely to skip sentences if they're long.
### Pictures are faster than words Consider using pictures to explain ideas. They get people's attention and can be much more efficient than words.
### Decide for me but let me have the final say Take your best guess and act rather than asking first. Too many choices and decisions make people unhappy. Just in case you get it wrong, allow for 'undo'.
### Only show what I need when I need it People get overwhelmed when they see too much at once. Break tasks and information into small, digestible chunks. Hide options that aren't essential at the moment, and teach people as they go.
### I should always know where I am Give people confidence that they know their way around.

Make places in your app look distinct and use transitions to show relationships among screens. Provide feedback on tasks in progress.
### Never lose my stuff Save what people took time to create and let them access it from anywhere. Remember settings, personal touches, and creations across phones, tablets, and computers. It makes upgrading the easiest thing in the world.
### If it looks the same, it should act the same Help people discern functional differences by making them visually distinct rather than subtle. Avoid modes, which are places that look similar but act differently on the same input.
## Only interrupt me if it's important Like a good personal assistant, shield people from unimportant minutiae. People want to stay focused, and unless it's critical and time-sensitive, an interruption can be taxing and frustrating.
## Make Me Amazing ### Give me tricks that work everywhere People feel great when they figure things out for themselves. Make your app easier to learn by leveraging visual patterns and muscle memory from other Android apps. For example, the swipe gesture may be a good navigational shortcut.
### It's not my fault Be gentle in how you prompt people to make corrections. They want to feel smart when they use your app. If something goes wrong, give clear recovery instructions but spare them the technical details. If you can fix it behind the scenes, even better.
### Sprinkle encouragement Break complex tasks into smaller steps that can be easily accomplished. Give feedback on actions, even if it's just a subtle glow.
### Do the heavy lifting for me Make novices feel like experts by enabling them to do things they never thought they could. For example, shortcuts that combine multiple photo effects can make amateur photographs look amazing in only a few steps.
### Make important things fast Not all actions are equal. Decide what's most important in your app and make it easy to find and fast to use, like the shutter button in a camera, or the pause button in a music player. # Can truly great design be done the open source way?
Source: OpenSource.com
A few weeks ago, I wrote an article about Apple and open innovation. The discussion in the comments about Apple's success, despite their non-openness, was pretty interesting. Greg DeKoenigsberg started things off with this salvo:
> "No community could build something as gorgeous as the iPhone; it requires the singular vision of a beautiful fascist, and the resources of a gigantic company, and a world full of users who would happily trade simplicity and certainty for the ability to tinker."
I think few people would argue that one of Apple's greatest strengths is their amazingly consistent, and consistently beautiful, design work. And when I say design, I mean both "little d design" (their stuff looks awesome) and "big D Design" (their systems, processes, and experiences are expertly rendered).
From a design perspective, Apple has figured out how to make lightning strike in the same place over and over again.
Today, I want to ask a question that I've been thinking about for a long time:
Can truly great design be done the open source way?
Meaning, can a group of people designing collaboratively, out in the open, ever do the kind of consistently beautiful design work that Apple does? Or is Greg right, that "no community could build something as gorgeous as the iPhone"?

Both of my partners at New Kind, David Burney and Matt Muñoz, are designers by background. Both of them have significant open source experience (David spent almost 5 years as the VP of Communications at Red Hat, Matt worked on many Red Hat projects, including designing the Fedora logo), so the three of us have talked about this subject many times before. Even on the Fedora logo project, which was one of our first experiments in open design (learn more about the project and process here), only the input and feedback part of the process was open. Ultimately the logo itself was designed by one person (Matt), with a lot of feedback along the way from a very engaged community.
I've spent time looking at "crowdsourced design" companies like 99designs and Crowdspring. Many folks have written about companies like this as the future of design (great post last week by Hutch Carpenter on crowdsourcing and the disruption of the design industry).
And old school designers are scared to death that these companies will commoditize the design industry and put them out of work. Maybe. But I'm not so sure.
Spend a few minutes on one of these sites and you'll see almost all of the projects are simple "little d design" projects: websites, t-shirts, icons, logos, that sort of thing. I didn't see a big market for experience design or other more complex "big D Design" activities.
These sites don't seem very collaborative to me either. Sure, you can see the designs that other users have submitted. It is transparent. But you are not incented to collaborate with other designers, almost the opposite, since only one design wins the money.
So tell me. Will great design ever be a bazaar, or will it always be a cathedral? Do you know of examples where amazing design work—big D or little d—is being done the open source way? Do you think open, collaborative design will ever be as good as (or better than) design done the traditional way?
I'm not sure.
I'd love to hear your thoughts. # Five steps to using design in your open source project
Source: OpenSource.com
At the Open Technology Institute (OTI), we've been working on opening our user feedback process as a way to improve our internal processes and collaboration, engage our user community more, promote non-developer contributions, and think more broadly about how open source process plays a role in the Commotion Wireless project, a free and open-source communication tool that uses mobile phones, computers, and other wireless devices to create decentralized mesh networks.
More on that in my previous article.
In this article, I've expanded on the five tips I've identified for leveraging user-centered design in your open source project.
1. **Track usability and design issues**.
- Gather context as well as feedback - Don't respond (with software/design changes) to each piece of feedback - Synthesize issues for themes
As a result of our field activities, we had a pretty large queue of feedback from trainers, trainees, and users on documentation, usability and design, feedback from hackathons, the developer and discussion mailing lists, as well as a tracker for issues that people submit via the website. Given the amount of feedback we had gathered over the last few years, we decided to start by synthesizing and categorizing the themes in that dataset.
Understanding user feedback can be very challenging. Many developers and designers hear

feedback or read notes from a user feedback session, and want to make changes to the user interface based on each comment a user makes. What a user says exactly is not always the change that you need to make—it's important to understand the context in which the user provided the feedback. We've run into this a few times on the Commotion project, as much of our feedback comes from hands-on training sessions with novice users.

The main problem here is that it becomes difficult to understand if the feedback is on the documentation that the user is following, something confusing in the interface design or a mismatch between the documentation and the interface. Understanding the context and the user motivations and goals, and documenting that with the actual feedback—observational or commentary—is crucial for assessing the feedback later. Not to mention that it makes it possible for other team members to understand the context without having been present.

1. **Understand who you want to design for**.

When making design decisions, you often have to prioritize which type of user to design for—novice or expert users. Our goal with the Commotion project is that anyone should be able to set up a mesh network, which would push us in the direction of designing for novice users.

The challenge is that in network administration and management, users will frequently and sometimes very quickly need access to the more advanced settings and configuration tools. Striking the balance between novice usability and expert access can be challenging, and it's important here to remember that each round of usability improvements is just that - the current round or iteration. You might not be able to fix everything this time, and you might not have even found all of the issues to fix.

The important thing is that you focus on keeping it simple and learnable to allow novice users to find their way around and expert users the ability or doors to get to the features they know they need.

3. **Make prototypes**.

Leverage the style of the prototype (low-fidelity vs high-fidelity) to get the type of feedback you need.

Similar to the issue I described earlier about understanding the context surrounding the user feedback, it's important to understand the context you are creating with your prototypes. High-fidelity prototypes or mockups are a great way to get detailed design/visual feedback from users—e.g. about colors, language, placement of buttons, etc. High-fidelity prototypes are not a good way to get feedback about the logical process flow—e.g. does your software match users' mental models of how the software should work?

The basic rule of thumb here is that the sketchier (literally more sketch-like) your prototypes are—the more likely the feedback will be about information structure and process flow—i.e. challenging your basic design assumptions, whereas the more detailed the prototypes are the less likely the user is to challenge it because they have a sense that you've already finished and so focus on the aesthetic issues. The type of prototype you make should be based on the type of feedback that you need.

4. **Understand how the software works from the user's perspective**.

The next step of the process for us was to reach out to user groups—some whom we had trained and some who had adopted the software on their own to have them do a walkthrough of the prototypes. We looked for novice users as well as expert users to make sure that the changes we were making wouldn't break anyone's mental models of the software and to make sure that we were burying any tools that needed to be top level in the interface. We scheduled as many users as we had time to work with, and used screen sharing software to walk through some standard tasks using Commotion.

To do this, we prepared a plan for our interactions with the user community, asking them to Think Aloud through some of the common tasks such as configuring a wireless router, checking on the network status, finding out how many users are currently connected, and things like that. We also had some standard "demographic" questions to get a sense of the user's context, such as their familiarity with the software and with managing networks, what type of network they were involved in, their role within the network, and what type of operating systems and technology they use most often. All of this information can help your team in building better user profiles or personas and can feed into more informed design processes for new features down the line.

5. **Use usability/design iterations to build your community: hackathons, mailing lists, and distributed tools**.

Similar to what the Magnolia CMS team described in their post about designing for mobile versus desktop, it’s important to be open and transparent about the usability review and design process as it can help you engage your broader community. This means cultivating conversations on mailing lists, encouraging constructive criticism of UX and design, or even asking current members to recruit others that work in UX to the project.

For the Commotion team, this was just one iteration of what has been and will continue to be many improvements to the user interface, but it gave us a chance to practice our process, catch up on a backlog of user interface issues, and work on developing our community. It also gave us an opportunity to show the community the many ways that they contribute to the process, aside from just contributing code. We’re making sure that whenever we go to a hackathon, we have our documentation ready to be hacked on along side our software, website, and everything else, so that we can keep making it better and easier to use.

# Leverage user-centered design in your open source project

Source: Opensource.com

When I first started working at the Open Technology Institute (OTI), I was consistently challenged with the question: "Why would a UX designer want to work at an open source organization?" The truth, in my opinion, is almost all design and usability work is by its nature open source.

Designers build on and adapt from design patterns that we see and experience in the world around us. We are constantly sharing and testing our ideas with our peers and collaboratively iterating on concepts because good design is widely understandable. There are great resources around the Internet [see below] on usability and design best practices, and many of the open source user interface frameworks are building these principles into their toolkits, so that the lines between developers, designers, etc are even more blurred.

The question remains though: Why do we so rarely see usability professionals and designers in the open source community, and how we might begin to change that?

Open source processes and projects are also not highlighted as frequently in design and UX schools and conferences, limiting the awareness in design communities of the needs in the field. Additionally, as I touched on above, the evolution of software frameworks, libraries, and toolkits, such as Twitter Bootstrap, and AngularJS are allowing the roles to blur, and making it easier to "fake it until you make it." While on the one hand it's become clear that usability and design are crucial needs for both user facing and developer facing projects (APIs need to be usable too!), there seems to be a divide as to the percentage of designers in proprietary versus open source software. In the past, small projects have viewed user experience (UX) as a luxury and therefore not invested in staffing their UX needs. Also, I imagine that the market demand for designers keeps many of them occupied in companies who can pay them competitively, where many (not all!) of open source projects are volunteer run, or operate on smaller budgets. However, I believe that one of the main reasons that we don't see as many designers and usability professionals in open source is that they aren't sure how to participate in projects and may not have had exposure to open source projects in the past.

At Open Technology Institute (OTI), we've been working on opening our user feedback process as a way to improve our internal processes and collaboration, engage our user community more, promote non-developer contributions, and think more broadly about how open source process plays a role in the Commotion Wireless project, a free and open-source communication tool that uses mobile phones, computers, and other wireless devices to create decentralized mesh networks.

For the official Version 1 release of the software, we decided to do a full overhaul of the user interface, focused primarily on the version for wireless routers, but also on aligning the other platforms (Mac, Linux, Android, Windows, iOS) to the router. Mesh networking firmware is not your everyday project—our development team frequently is focused on how to add more functionality while simultaneously making the files smaller so that they can fit on hardware with very little space (only 5.2 MB!). Meanwhile our field team, of which I am part, is researching how mesh networks can leverage existing social relationships, developing training and adoption frameworks, and testing these ideas out with communities who are interested in experimenting with community-owned infrastructure.

In my next article, I will expand on these 5 tips that we learned from our user-centered design review of the Commotion interface.

5 tips for leveraging user-centered design in your open source project

1. **Track usability and design issues**. Gather context as well as feedback. Don't respond (with software/design changes) to each piece of feedback. Synthesize issues for themes.
2. **Understand who you want to design for**. When making design decisions, you often have to prioritize which type of user to design for—novice or expert users.
3. **Make prototypes**. Leverage the style of the prototype (low-fidelity vs high-fidelity) to get the type of feedback you need.
4. **Understand how the software works from the user's perspective**.
5. **Use usability/design iterations to build your community: hackathons, mailing lists, and distributed tools**.

### Additional Resources
- 10 Usability Heuristics - How to write a good UX Bug report - What UX Methods To Use And When To Use Them - Usability Testing Methods - Low Cost User Experience Testing - The Encyclopedia of Human Computer Interaction, 2nd Ed # Making Security Boring

Source: Guardian Project

We have been thinking about how to approach the next five years of our work. An idea of "security so easy and seamless, that it is boring" came to the surface through some discussions. This led us to look for inspiration in important inventions and innovations of the past, that

provide safety and security to all on a day-to-day basis, without the users of these technologies hardly thinking about them. This is no longer about James Bond super-spy technologies, it is about having as little impact on your day-to-day use of mobile technology while still providing the maximum protection to your data and communications, as possible.
Here then, as inspiration and our guiding lights, is our list of safety inventions of the past that we aim to be "Boring Like…."
IMAGE - Old-seatbelt-image
Boring Like Seat Belts (buckle up!)
IMAGE -
StateLibQld_1_212036_Cream_pasteurising_and_cooling_coils_at_Murgon_Butter_Factory,_1939
Boring Like Pasteurized Milk
IMAGE - versagate3
Boring Like Baby Gates (kids and dogs hate 'em, but they are a necessary fun killer for any household)
IMAGE - smokealarm
Boring Like Smoke Detectors (though our friends at Birdi are aiming to make them exciting again…)
IMAGE - childlids
Boring Like Childproof Lids (again, for the parents out there, a literal life saver)
IMAGE - sft_08_01
Boring Like Anti-Lock Brakes (the modern automobile is one of the best examples of bore-sec!)
IMAGE - bg_SurgeProtectorBuyingGuide_hero_image
Boring Like Surge Protectors (wouldn't you rather lose your $2 power strip then your $1000 computer?)
IMAGE - RestaurantInspectionGradeA
Boring Like Restaurant Inspections (peace of mind from a single letter rating when you go out to eat…)
IMAGE - dougie_2008_in_his_outwardhound_dog_life_jacket
Boring Like Life Vests (even dogs love 'em!)
With our work on critical security and privacy enhancements for mobile devices and apps, we aim to bore. The best security is the kind you don't have to worry about, until you need it (and then you won't know how you ever lived without it…)
Here's to breaking new "Bore-Sec" ground in 2015! Happy New Year!
# Design and Usability
## About this section Design and usability are two critical components often overlooked. This short section will provide you a high-level view of how user centered design improves usability.
## Articles - Intro to Android design principles - Making security boring - Can truly great design be done the open source way? - Where design thinking and open source community collaboration meet - 5 tips: Leverage user-centered design in your open source project - Five steps to using design in your open source project
# Where design thinking and open source community collaboration meet
Source: OpenSource.com
In comparing traits associated with design thinking collaboration and collaboration in the open

source community, there are many parallels: open exchange, broad participation, rapid prototyping.
There's also one really interesting contrast: The mindset you tend to see when generating and choosing ideas. But what I'll suggest here is that when you apply the best elements of these two mindsets at just the right time in their respective processes, the results can be pretty amazing.
I'll start with design thinking.
The brainstorming phase of design thinking is probably the most recognizable part of the process. These brainstorming sessions are designed to be open. Lots of ideas are generated, ideas are built on ideas--there are no bad ones. To do that, we have to establish ground rules. Stanford's d.school posts their list, and our rules at Red Hat have many similarities. Rather than criticizing someone's ideas, you have to build on them and suggest your own. No devil's advocates allowed. In this environment, a good collaborator is one who generates and encourages the most ideas from others.
There are a few good reasons for the rules. First, criticizing an idea too soon can kill that idea in its most fragile state. It doesn't give others a chance to carry it forward and improve on it. Second, even the wildest--sometimes worst--ideas could spark a great idea in someone else. This happens all the time. Often the worst ideas and best ideas are closer than they seem.
And third, it's essential to create an open, positive environment so people are willing to share their ideas and explore. You've got to believe the best idea is always right on the verge of discovery.
Community collaboration is similarly open, but can be somewhat less nurturing.
A solution is found to have merit when it can stand up to scrutiny. In other words, what doesn't kill your idea makes it stronger.
Here a good collaborator is often the one who finds the most bugs. So rather than looking for the positive aspects to build upon we look for holes. And that's a good thing. As the classic open source saying goes, "With many eyes, all bugs are shallow." It's better to find them in beta.
But that doesn't make it easy. In fact, sometimes it can feel like a middle school dodgeball game where you put yourself up against the wall and invite your classmates to take their best shot.
There are good reasons why this environment works this way, too. If a solution is going to fail, it needs to fail early so it can fail forward. A tough testing environment isn't going to get easier in the real world.
Second, in open source the code is shared, so there's always a possibility, and even an expectation, for improvement. A solution can always be better. If you're lucky, you'll find lots of people to tell you so. If you're really lucky, they'll offer to help fix it. That's the beauty of the model.
In both cases, each environment is dependent on an open exchange. Everyone has access to information. Many people participate. Even design thinking processes have a prototype stage.
But even though their mindsets may seem in conflict with each other--I believe they're two important functions of the same process, simply applied at different stages:
When you're in the early stages of brainstorming and generating ideas--take care to protect creative thinking and the ideas they generate so those ideas can multiply.
And when you've got that idea you think is good--then it's time to to put it to the test. Find the holes. Invite smart people to do the same. I'll be the first to admit this is far easier said than done. Sometimes it can be nearly impossible. But if the idea is still standing at the end, it'll be

all the better for it. And then you know you've really got something. # Building Apps for a Complex World
something here about why we need security, etc.
The idea of an application is simple. The world where that application lives is not. To be a *great* software developer, one has to manage the following:
- Usability and design - Practicality and devices - Security and infrastructure
(*TBD*...) # The Elements Of The Mobile User Experience
Source: Smashing Magazine
## Introduction Mobile users and mobile usage are growing. With more users doing more on mobile, the spotlight is on how to improve the individual elements that together create the mobile user experience.
The mobile user experience encompasses the user’s perceptions and feelings before, during and after their interaction with your mobile presence — be it through a browser or an app — using a mobile device that could lie anywhere on the continuum from low-end feature phone to high-definition tablet.
Creating mobile user experiences that delight users forces us to rethink a lot of what we have taken for granted so far with desktop design. It is complicated in part by mobile-specific considerations that go hand in hand with small screens, wide variations in device features, constraints in usage and connectivity, and the hard-to-identify-but-ever-changing mobile context.
Dissecting the mobile user experience into its key components gives us a conceptual framework for building and evaluating good mobile experiences, within the context of a user-centered approach to designing for mobile. These components shape the mobile user experience — including functionality, context, user input, content and marketing, among others.
### The elements of mobile user experience * Functionality * Information Architecture * Content * Design * User Input * Mobile Context * Usability * Trustworthiness * Feedback * Help * Social * Marketing
The relevance of these elements will change depending on the type of device (feature phone versus smartphone versus tablet) and the presentation interface (app versus Web). This article briefly describes each of these elements and elaborates on each with selected guidelines.
## Functionality This has to do with tools and features that enable users to complete tasks and achieve their goals.
### GUIDELINES - Prioritize and present core features from other channels that have especial relevance in a mobile environment. For an airline, this includes flight statuses and flight check-ins. For cosmetic chain Sephora, it includes supporting in-store shopping via easy access to product reviews on mobile devices. - Offer relevant mobile-only functionality (like barcode scanning and image recognition), and enhance functionality using the capabilities of mobile devices where possible to engage and delight users. Old Navy’s app serves up surprise games or savings when users snap the logo in a store. - Ensure that fundamental features and content are optimized for mobile. For example, make sure the store locator shows the nearest stores based on the device’s location, and make the phone numbers click-to-call. - Include features that are relevant to the business category. For retail websites and apps, this would include product search, order status and shopping cart. - Offer key capabilities across all channels. Users who sign in should see their personalized settings, irrespective of the device or channel being used. If

certain functionality is not offered on mobile, then direct users to the appropriate channel, as TripIt does to set up a personal network. TripIt directs users to the website for setting up a network

### ADDITIONAL READING "Sharing App Bump 3.0 Slashes Most Features, Proves Less Really Can Be More," Fast Company

## Information Architecture This has to do with arranging the functionality and content into a logical structure to help users find information and complete tasks. This includes navigation, search and labeling.

### GUIDELINES - Present links to the main features and content on the landing page, prioritized according to the user's needs. Mobile Design Pattern Gallery has examples of primary and secondary navigation patterns for mobile, many of which are vertical instead of horizontal as on desktop websites. - Enable mobile users to navigate to the most important content and functionality in as few taps or key presses as possible. Navigation optimized for small screens is usually broad and shallow instead of deep. While three clicks (or taps) is not the magic number, users need to be able to recognize that each tap is helping them complete their task. Every additional level also means more taps, more waiting for a page to load and more bandwidth consumed. - Address the navigation needs of both touchscreen and non-touchscreen users. When designing for touch, make sure the tap size of the navigation item is at least 30 pixels wide or tall. Provide keypad shortcuts for feature phones, so that users can enter, say, a number (0 to 9) to quickly access a link: - Cater to feature phone users, as CNN does with access keys, not as Delta does by making the first action to be nine key presses downs. - Provide navigational cues to let users know where they are, how to get back and how to jump back to the start. Mobile breadcrumbs are often implemented by replacing the "Back" button with a label showing users the section or category that they came from. For mobile websites, use standard conventions, such as a home icon that links back to the start screen, especially when navigation is not repeated on every screen. - Use concise, clear, consistent and descriptive labels for navigation items and links. While always a good practice, it becomes even more important on tiny mobile devices.

### ADDITIONAL READING - "Chapter 1: Navigation," Mobile Design Pattern Gallery, Theresa Neil

## Content Otherwise known as "the stuff on your website" (as Lou Rosenfeld and Peter Morville refer to it in *Information Architecture for the World Wide Web*), content is the various types of material in different formats, such as text, images and video, that provide information to the user.

### GUIDELINES - Present an appropriate and balanced mix of content to users (product information, social content, instructional and support content, marketing content). - Use multimedia when it supports the user's tasks in a mobile context, adds value to the content or supports the goals of the website. Most of the time, multimedia content is best provided when the user is looking for distraction or entertainment (such as news or funny clips) or when it has instructional value (for example, how to use an app or new feature). - Always give the user control over multimedia content by not auto-starting video or sound, by allowing the user to skip or stop multimedia content and by being mindful of the bandwidth it takes up. - Ensure that content is mobile appropriate. Just as we had chunking guidelines when going from print to Web, copy should be written for shorter attention spans on mobile devices. Optimize images and

media for the device; this means scaling down for smaller devices and making sure images are sharp enough for the new iPad. - Ensure that primary content is presented in a format supported on the target device. Even now, websites such as Volkswagen’s ask iOS users to download Flash. - VW asks iPad users to download an unsupported Flash plugin
### ADDITIONAL READING - “Mobile Content: If in Doubt, Leave It Out,” Jakob Nielsen, Alertbox - “Mobile Content Strategy Link-o-Rama 2011,” Karen McGrane
## Design This has to do with the visual presentation and interactive experience of mobile, including graphic design, branding and layout.
### GUIDELINES - Remember the sayings “Mobilize, don’t miniaturize” (popularized by Barbara Ballard) and “Don’t shrink, rethink” (of Nokia). Both make the point that mobile design should not just rehash the desktop design. - Design for glance ability and quick scanning. Glance ability refers to how quickly and easily the visual design conveys information. - Maintain visual consistency with other touchpoints and experiences (mobile, app, Web, print and real world) through the use of color, typography and personality. Identifying Amazon in the stack below is easy even though the brand name is not visible. - Guide users from the initial and most prominent element of the design to other elements to help them complete their tasks. This is known as visual flow. A good design brings together visual elements as well as information architecture, content and functionality to convey the brand’s identity and guide the user. - Consider both portrait and landscape orientations in the design process. Devices increasingly support multiple orientations and automatically adjust to match their physical orientation. Maintain the user’s location on the page when they change orientation. Indicate additional or different functionality in the new orientation if applicable. For example, the ING app informs users about additional features in the landscape mode
### ADDITIONAL READING - “Designing Glanceable Peripheral Displays” (6 MB, PDF), Matthews, Forlizzi and Rohrbach, UC Berkeley - Universal Principles of Design, Revised and Updated: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design, William Lidwell, Kritina Holden, Jill Butler
## User Input This has to do with the effort required to enter data, which should be minimized on mobile devices and not require the use of both hands.
### GUIDELINES - Limit input to essential fields. Or, as Luke Wroblewski says in his book *Mobile First*, “When it comes to mobile forms, be brutally efficient and trim, trim, trim.” Limit registration forms to the minimum fields required, and use shorter alternatives where possible, such as a ZIP code instead of city and state. My favorite offender of this guideline is Volkswagen’s form to schedule a test drive; the mobile form has more required fields than the desktop version. - Display default values wherever possible. This could be the last item selected by the user (such as an airport or train station) or the most frequently selected item (such as today’s date when checking a flight’s status). - Offer alternate input mechanisms based on the device’s capabilities where possible. Apps take advantage of quite a few input mechanisms built into devices, including motion, camera, gyroscope and voice, but mobile websites are just starting to use some of these features, particularly geolocation. - Use the appropriate input mechanism and display the appropriate touch keyboard to save users from having to navigate their keyboard screens to enter data. Keep in mind that inputting data is more tedious on feature phones that have only a numeric keypad. For non-sensitive applications, allow users to stay signed in on their mobile device; and save information such as email address and user name

because mobile phones tend to be personal devices, unlike tablets, which tend to be shared between multiple people. - Consider offering auto-completion, spellcheck suggestions and prediction technology to reduce the effort required to input data and to reduce errors — with the ability to revert as needed. Disable features such as CAPTCHA where not appropriate.
### ADDITIONAL READING "Forms on Mobile Devices: Modern Solutions," Luke Wroblewski
## Mobile Context A mobile device can be used at anytime, anywhere. The mobile context is about the environment and circumstances of usage — anything that affects the interaction between the user and the interface, which is especially important for mobile because the context can change constantly and rapidly. While we often focus on distractions, multitasking, motion, low lighting conditions and poor connectivity, it also includes the other extreme — think using a tablet in a relaxed setting over a fast Wi-Fi connection.
"The Context of Mobile Interaction," Nadav Savio
### GUIDELINES - Use device features and capabilities to anticipate and support the user's context of use. The iCookbook app allows users to walk through a recipe using voice commands — a nice feature when your hands are covered in batter! - Accommodate for changes in context based on the time of day and when the user is using the app. The Navfree GPS app automatically switches from day to night mode, showing low-glare maps for safer nighttime driving. - Use location to identify where the user is and to display relevant nearby content and offers. A Google search for "movies" on a mobile device brings up movies playing nearby and that day's showtimes, with links to buy tickets online if available. - Leverage information that the user has provided, and respect their preferences and settings. After the first leg of a multi-leg flight, TripIt showed me the flight and gate information for my next flight, as well as how much time I had to kill. United's app did no such thing, even though it knew much more about me. It could have shown me how to get from my current plane to the connecting flight and highlighted the location of the United Club along the way, where I could comfortably spend my two-hour wait, since it knew I was a member. - Default to the user experience most appropriate for the device (i.e. a mobile experience for small screens, and perhaps a desktop-like experience for tablets), but give users the option to have enhanced features. A big discussion on how to present this to the user recently took place, with Jakob Nielsen recommending a separate mobile website and Josh Clark arguing instead for a responsive design; yet others believe that Nielsen and Clark are both wrong.
### ADDITIONAL READING - "The Context of Mobile Interaction" (0.2 MB, PDF), Nadav Savio and Jared Braiterman - "On Mobile Context," Jason Grigsby Don't miss the links to resources on the mobile context near the end. - "When and Where Are People Using Mobile Devices?," Luke Wroblewski
## Usability This is the overall measure of how well the information architecture, design, content and other elements work together to enable users to accomplish their goals.
### GUIDELINES - Make it clear to the user what can be selected, tapped or swiped (this is known as affordance), especially on touchscreen devices. One of the big findings of Nielsen Norman Group's usability studies of the iPad was that users didn't know what was touchable or tappable. Another issue was swipe ambiguity: when the same swipe gesture means different things in different areas of a screen. Ensure that touchability is clear and that items such as links, icons and buttons are visibly tappable. - For touchscreen devices, ensure that touch targets

are appropriately sized and well spaced to avoid selection errors. Also, place touch targets in the appropriate screen zones; for example, put destructive actions such as those for deletion in the "Reach" zone, as shown by Luke Wroblewski in his book *Mobile First*. - Follow conventions and patterns to reduce the learning curve for users and to make the mobile experience more intuitive. Dedicated apps should follow platform-specific standards and guidelines. A comprehensive collection of links to official UI and UX guidelines is available in the article "UI Guidelines for Mobile and Tablet Web App Design" on Breaking the Mobile Web. - Ensure usability in variable conditions, including for daylight glare and changed angle of viewing and orientation, by paying attention to design elements like contrast, color, typography and font size. - Do not rely on technology that is not universally supported by your audience's devices, including Java, JavaScript, cookies, Flash, frames, pop-ups and auto-refreshing. When opening new windows or transitioning from an app to the browser, warn users to avoid overwriting already open tabs.
### ADDITIONAL READING - "Usability of iPad Apps and Websites," Nielsen Norman Group - "UI Guidelines for Mobile and Tablet Web App Design," Max Firtman - "Mobile Usability Update," Jakob Nielsen, Alertbox
## Trustworthiness This relates to the level of confidence, trust and comfort that users feel when using a mobile website or app. According to a 2011 study by Truste and Harris Interactive, privacy and security are the top two concerns among smartphone users: - **Privacy and security are the top two concerns among smartphone users**.
### GUIDELINES - Do not collect or use personal information (such as location and contact list) from mobile devices without the explicit permission of the user. The first few months of this year have seen numerous reports of apps secretly copying smartphone address books, with watchdogs up in arms and users retaliating. - Make it easy for users to control how their personal information is shared in a mobile app by asking before collecting their location data and by allowing them to opt out of targeted advertising. - Clearly state your business practices (including for privacy, security and returns), and present them contextually (such as by displaying links to your privacy and security policies on the registration screen). The policies themselves should be accessible in a secondary section of the mobile user experience (such as the footer or a "More" tab). Reinforce credibility by displaying trusted badges, especially when users need to trust you with their personal or financial information. - Present policies appropriately on mobile devices by offering a concise summary and an option to email the entire policy. Privacy and security policies tend to be notoriously long and full of boring legalese that users often blindly click through to continue what they really want to do, so make it easy for users who are interested in the fine print. - Don't break the user's workflow when displaying legalese. Take them back to where they were before being interrupted, instead of making them start all over.
### ADDITIONAL READING - "Layered Policy Design", TRUSTe Blog
## Feedback This has to do with the methods for attracting the user's attention and displaying important information.
### GUIDELINES - Minimize the number of alerts the app displays, and ensure that each alert offers critical information and useful choices. For a smile, look at Chris Crutchfield's video on notification and alert overload. - Keep alerts brief and clear, explaining what caused the alert and what the user can do, along with clearly labeled buttons. - Notifications should be brief and

informative, not interfere with anything the user is doing, and be easy to act on or dismiss. - Provide feedback and confirmation on screen without disrupting the user's workflow. - If your app displays badges and status bar notifications, keep the badges updated and clear them only when the user has attended to the new information. Chase clears the notifications badge for its mobile app the moment the user visits the notification section, even before the user has seen which of their multiple accounts triggered the badge, forcing them to hunt through each account to see what triggered it.
### ADDITIONAL READING - "Mobile Notifications," Fred Wilson - "The Future of Mobile Notifications," True Ventures - "Chapter 8: Feedback and Affordance," Mobile Design Pattern Gallery, Theresa Neil
## Help This relates to the options, products and services that are available to assist the user in using the website or app.
### GUIDELINES - Make it easy for users to access help and support options. Users commonly look for help in the footer of a mobile website and in the toolbar or tab bar of an app. - Offer multiple ways to get support, including options relevant in a mobile context, such as self-serve FAQs, live support via click-to-call, and near-real-time Direct Message tweets. Two financial service companies that actively offer support via Twitter are American Express and Citibank. - Present a quick introduction and short tutorial on using the app when it first launches, with options for the user to skip and view later. - When introducing new or unique functionality (such as when check depositing via mobile apps was first introduced), offer contextual help and tips to guide users the first time, and as a refresher for infrequently used functionality. - Offer help videos when appropriate, but allow the user to start, pause, stop and control the volume as they wish, and keep in mind the multimedia guidelines mentioned in the "Content" section above.
### ADDITIONAL READING - "Chapter 7: Invitations" and "Chapter 9: Help," Mobile Design Pattern Gallery: UI Patters for Mobile Applications, Theresa Neil. Chapter 7 is available online at UX Booth. - "Top 6 Help Design Patterns for iPhone Apps," Catriona Cornett, inspireUX
## Social This relates to content and features that create a sense of social participation, that enable user interaction and that facilitate sharing on established social networks.
### GUIDELINES - Create and maintain a presence on social networks (for example, a Facebook page) and local services (for example, a profile page on services such as Google Places, Bing Business Portal and Yahoo Local). These will be highlighted in search results and on location-based social networking services. In addition to your business' name, include your physical address, phone number, URL and hours of operation. - Incorporate your social presence and activity into your website's mobile experience by showing your recent activity and offering an easy way to follow or like you on these networks. - Integrate social networking features into your website's mobile experience to make it easy for users to connect with their own social networks. This could be as simple as using APIs to enable social sharing, bookmarking, tagging, liking and commenting. - Invite users to generate content featuring your brand, product or service from their mobile device, offering some incentive in return. For example, the burger chain Red Robin could invite the user to share a picture of their child reading a school book at one of its locations to get a free milkshake. - Provide mobile offers that can be shared and go viral. American Express currently offers savings and discounts to users who sync their profiles on networks such as Facebook, Twitter and Foursquare to their credit card. - Apps that rely on

social contributions from users should look at ways to seed content in a way that is useful and, eventually, self-sustaining. For example, the My TSA app has a user-contributed feature that shows the wait times at security checkpoints, but it often shows outdated information, even though airport staff post physical signs of wait times at some airports.
### ADDITIONAL READING "The Definitive Guide to Adding Social Features to Your Mobile Apps", Verious
## Marketing This has to do with the methods by which a user finds a website or app and the factors that encourage repeated usage.
### GUIDELINES - Ensure findability by optimizing for mobile search and discovery, such as by keeping URLs short. If you have a separate mobile website, follow URL naming conventions (`m.site.com` or `mobile.site.com`). In mobile search results, provide quick access to location-based content (e.g. directions from one's current location) and device-formatted options (e.g. click to call). - "Quick response" (QR) codes should lead to a mobile-optimized landing page, instead of a traditional page that requires zooming or, worse still, to the website's home page, from where the user has to hunt for information. As a side note, QR codes painted on buildings should be big and clear enough to be recognized and deciphered by mobile devices. - Email campaigns should include a link to view the message in a mobile-friendly format, which itself links to the relevant offer page formatted for mobile — unlike CVS/pharmacy, which takes users to its mobile home page. - Promote your app in other channels where possible (TV, print and in-store advertising), and offer incentives to download and use the app, usually in the form of discounts and savings. If your app has a price tag, attract users to buy it in an overcrowded market by offering a limited-time promotional price. Another option is to promote the app through the Free App A Day marketplace. - Prompt users to rate and review your app or to share it on social networks after they have used it, but give them the option to postpone or stop these prompts. This will not only generate word of mouth, but give you insight into what users like and don't like about the app. "Taking Control of Your Reviews" by small tech discusses the strategy of encouraging happy customers to post reviews and unhappy customers to email you feedback.
### ADDITIONAL READING - "iPad App Marketing Case Study: Flickpad," Chad Podoski, Mobile Orchard - "The Art of Launching an App: A Case Study," John Casey, Smashing Magazine - "How to Market Your Mobile Application," Michael Flarup, Smashing Magazine
## Conclusion Mobile user experience is still a developing field, and opportunities for improvement continue to emerge. We've presented an overview of the key elements of the mobile user experience, along with some guidelines to get started in each. Focusing on these individual elements will help us create great overall mobile user experiences for our users. # Getting started in Android Development
Original Title - *I Want to Write Android Apps. Where Do I Start?* Source lifehacker
> Dear Lifehacker, > I have some background in coding, but I've never touched Android development before. I'd like to get started, but I'm not entirely sure what I need. I don't need to "learn to code" per se, but I could use some guidance on where to start with Android. Can you help?
> Sincerely, > Dreaming of Electric Sheep

Dear Mr. K. Dick, As you're probably aware, writing apps for Android is more than just learning code syntax. If you've never learned to code, you can check out plenty of resources here.

However, there are still a whole host of tools and resources you might not be familiar with that you may need to make Android apps.

Note: this is not meant to be a comprehensive guide on every detail of these applications and resources. In fact, such a guide could more accurately be described as a book. However, we will give you an overview of the different tools you can use and where to find more information.

These tools require varying levels of experience and if you've never touched code before, you might want to check out our guides linked here. However if you're ready to move from theory and syntax to actual development, here's what you'll need.

## The Android Software Development Kit (or SDK)

The Android Software Development Kit (SDK) is actually a collection of tools that will help you make Android apps. There's more outside the SDK that we'll discuss, but here are some of the most helpful tools in the SDK:

### Eclipse/Android Studio

There are two primary integrated development environments (IDE) for Android. An IDE is the main program where you'll write code and put your app together. It can help you organize and edit the various files in your app, manage the packages and supporting libraries you app will need, and test it out on real devices or emulators.

The default IDE for Android is Eclipse. Eclipse allows you to modify Java and XML files and organize the various pieces of your application, among many other tasks. The version you get from Google also includes a package manager that allows you to update to the latest version of Android tools as soon as Google releases them.

The main alternative is Android Studio, which is currently being made directly by Google. Like many Google projects, Android Studio is part of a prolonged beta. The long-term intention is for Android Studio to replace Eclipse as the primary IDE for Android development. That doesn't necessarily mean it's for everyone. For example, if you need to make use of the Native Development Kit for apps like games (hint: if you need it, you probably already know you need it), Eclipse is mandatory. However, Android Studio is a good option if you want to get a jump start on the future, and you're willing to tolerate some possible bugs.

No matter which IDE you choose, using it is a bit like Photoshop: it can do a ton of cool things, but you'll probably only learn the individual tools as you need them. However, this is also a good place to get started on some of the basics of Android development. Here are some great tutorials and resources to get you started:

* Udacity - Developing Android Apps: This 8-week online class has a good amount of free elements, taught directly by Google engineers. The course won't just copy-paste code, but it will help you learn some of the core concepts and features you'll need. * Android Developer Training: Part of Google's documentation includes training tutorials on how to use its tools. These documents will walk you through basic features of the IDE. If you don't have much experience developing applications, this might not turn you into a master dev, but it will help you learn the tools. * Vogella: It's worth mentioning Vogella tutorials in just about every section here. This massive set of tutorials covers just about everything you could cover. If you have a basic question not covered above, check Vogella.

### ADB

We've talked about ADB before from a regular user perspective, but the tool's primary purpose is actually to aid in development. As such, it's included in the Android SDK. You can use this to

load software or make changes to your devices when it's plugged into your computer. Here are some of the basic tools you can use with ADB, but if you want to learn more as a developer, check these out:
* ADB Documentation: This is the primary resource from Google on what ADB is and how it works. You can find most of what ADB is capable of here. * Vogella - Using the Android Debug Bridge: Another Vogella tutorial, this one covers the basics of how ADB works and some of the common things you can do with it. If you don't want to dig through Google's documentation for the one command you need, this might be a good place to start.
## Android Developer Guidelines
We've already linked to a couple of resources from the official Android Developer Guidelines so far, which only proves how useful they are. Google maintains a vast, extensive collection of documentation and resources for how to program your apps that you can reference or search through.
If you're brand new to Android development, it can't hurt to browse through some of the tutorials and guides here. They're laid out in such a way that one lends into another (see the Android Developer Training above). Here are some sections that are worth brushing up on if you're getting started:
* **Google Services**: We've talked about Google Play Services before, but here's where you get to see what's going on under the hood. Google offers a wide variety of features that you might otherwise have to build out yourself like map and location features, cloud backups, sign-in services and more. You can check them all out here. * **API Guides**: Google services are set apart from the regular APIs, which you can also read about here. These range from code to create basic animations, to reading sensors and connecting to the internet. There's tons of info here to add functionality to your app. * **Sample Code**: Sometimes it helps to see how someone else did it before you. This section shows you samples of code for various functions. This can help you see how something works, or just use it in your app so you don't have to reinvent the wheel. Android Design Guidelines
The counterpart to the developer guidelines is the Design Guidelines. Google is focusing increasingly on teaching its developers how to make apps that not only work well but look good. As such, that means a lot of the work has been done for you to cover the basics like buttons, simple animations, and whatnot.
The place to go to get more info on this is the Android Design Guidelines, which are a second major subsection of Google's official documentation. Keep in mind that these are here for people who may not have a great grasp on visual design as it relates to creating application interfaces. In other words, if you already know what your app is going to look like, you might not need this. If you already know what you're app looks like but you're not good at making apps look good, check this out.
Here are a list of some the helpful areas to start:
* **Devices**: Android targets more than just phones. This section will help you learn how phones, tablets, TVs, and watches all relate and how you can design an interface that adapts to all of them. * **Patterns**: Android is built on structured interfaces. This section teaches the building blocks of how apps work so you can design the framework that you'll be building your design on top of. * **Material Design Documentation**: This is technically a separate section for now, but Google's newest version of Android will introduce a new type of design language called

Material Design. Here you can peruse what that means and how to think about designing apps that fit these guidelines. It's also helpful if you're not experienced with thinking about how users interact with apps, even if you don't follow the specific recommendations.
## GitHub/BitBucket
While you're developing an app, there are a lot of files to manage and you'll need a way to track changes. Git is one of the most commonly used protocols to manage new versions or changes to existing software. Necessarily, it's a little more complicated than a basic backup tool. It's flexible enough to allow you to manage multiple different branches of your app as well as pull from older versions if something goes wrong.
Two of the most common services for managing projects with Git are Github and Bitbucket. Both use the same underlying protocol and can be integrated directly into either Eclipse or Android Studio. BitBucket allows you to have some private repositories (read: storage for projects) without paying money, while GitHub's free offerings require them to be publicly listed unless you pay a little extra. Here are some resources that can help you get started with Git:
* **BitBucket Tutorials**: Atlassian, the maker of BitBucket, have a series of guides on how to get started with BitBucket and import your projects here. In my personal experience setting up both BitBucket and GitHub, this service and these guides were much easier for the uninitiated to get started with. * **GitHub Guides**: GitHub similarly has some tutorials on how to set up its service that you can find here. Some of the guides refer to older versions of the software in some cases, but generally you should be able to get up and running with these. * **Vogella Git Tutorial**: Vogella has yet another great tutorial here explaining what Git itself is and how it can help you manage your entire project. While version management is Git's primary function, there's a lot more here that Vogella can walk you through.
Developing for Android is far more than just putting Java in a text editor. If you have a little bit of experience with writing code but haven't dived head first into actual app development yet, there's a lot you may not be aware you need to know just yet. The good news is, you're not the first person to go down this road. These are just some of the tools you need and hopefully these guides will put you on the right path.
Sincerely, Lifehacker
# Application Fundamentals
- Elements of the mobile user experience - Building apps for a complex world - Getting started in android development
# Encrypt all the bits - introduction to mobile security
This is a paraphrase of Encrypt all the bits presentation.
## Intention vs Execution
## Session Overview - Overview of Guardian Project Apps & Developer Libraries (30m) - Threat Models and War Stories: Open Discussion about Risks, Fears and Security Needs (30m) - Encrypted Databases: securing structured data in activities, services and content providers (1hr) - Encrypted Files: securing arbitrary files from small to large (30m) - Secured Networking: defending against man-in-the-middle, SSL stripping, filtering and more (30m) - Hands-On Implementation time for sample work or debugging your own apps with new security features (1.5hr)
## Encryption a very quick introduction
### What is Encryption? - Plaintext + Algorithm + Key =Ciphertext - Symmetric vs

Asymmetric, Private vs Public - Randomness: Actual vs Pseudo - Common Cryptography Tools: OpenSSL, PGP (GnuPG!), BouncyCastle
### Android Built-in Encryption - HTTPS / TLS / SSL - javax.crypto “BouncyCastle” - OpenSSL - Full Disk Encryption - Android KeyChain (> API 18)
### CipherKit https://guardianproject.info/code
### CipherKit “Platform” Insert image here
### "CipherKit" Dev Libraries CipherKit is designed for Android app developers to make apps that are able to ensure better privacy, security and anonymity
#### SQLCipher: Encrypted Database SQLCipher is an SQLite extension that provides transparent 256-bit AES encryption of database files. It mirrors the standard android.database API. Pages are encrypted before being written to disk and are decrypted when read back.
#### IOCipher: Encrypted Virtual Disk IOCipher is a virtual encrypted disk for apps without requiring the device to be rooted. It uses a clone of the standard java.io API for working with files. Just password handling & opening the virtual disk are what stand between developers and fully encrypted file storage. It is based on libsqlfs and SQLCipher.
#### NetCipher: Encrypted Network Data & Tor Integration NetCipher is improving network security. It provides a strong TLS/SSL verifier to help mitigate weaknesses in the certificate authority system. It eases the implementation of supporting SOCKS and HTTP proxies into applications and also supports onion routing for anonymity and traffic surveillance circumvention.
### Let’s take a step back... (to figure out what it is we are worried about)
## Basic Threat Modeling * “What are you worried about?” aka Possible Attack Vectors * What data are you collecting or services are you providing that might be enticing or exposed? * Are the potential threats you face coming from the device (other apps or physical access) or the network?
## War Stories? * Have your apps, your business or your users or customers lives or businesses been affected by malware or security breaches? * Do you work in an industry that has specific requirements related to security and privacy? * Do you target a region of the world where users might be more exposed to attack, surveillance or privacy violations?
## Threat Landscape * Forensic Analysis * Rooting / Jail breaking * OS Issues * Infrequent Updates * Removable Storage * Cloud Services * Targeted Attacks * Device Sharing
### Malware is on the rise Malware on the rise: http://blog.trendmicro.com/trendlabs-security-intelligence/mobile-malware-high-risk-apps-hit-1m-mark/
### Cached GPS data stored in plain text http://elifelog.org/book/iphone-gps-cache-data
### Forensic Extraction "Universal Forensic Extraction Devices" can quickly and easily copy all of the data from a mobile phone.
If tools like these fall into the wrong hands, it is easy to assume any unencrypted data on a device can be easily stolen.
http://www.cellebrite.com/mobile-forensics
### Man in the middle Man-in-the-Middle: http://thehackernews.com/2013/03/t-mobile-wi-fi-calling-app-vulnerable.html
## Trust Landscape
| ID | Name | Description | | ---- | ---- | ---- | | 1 | Owner of the mobile phone | The primary operator of the mobile device. Assumed to have full access to the device, potentially secured

with a PIN/password screen. | | 2 | Detainer / criminal / bad actor | An authority figure or criminal who has or will be detaining the Owner[1]; has access to mobile phone. may have only manual/brute force access, or could have more sophisticated forensic extraction tools. | | 3 | Operator of the mobile network | Access to call and message logs (sender/receiver/message content) and cell tower association data (rough location) | | 4 | Employer, family or support organization; | May know the Owner[1]'s PIN/password, but otherwise has no access to data or network information; On the receiving end of an emergency message | | 5 | Malicious App / Backdoor / Malware / Forensics App | Access to some or all of the the Owner[1]'s data depending upon app data permissions and encryption, as well as how full the backdoor is. Authorization is often required by the user to allow apps to access data. |

## Assets

| ID | Name | Description | Trust Level | | ---- | ---- | ---- | --- | | 1 | Personal Data | Names, emails, phone numbers, calendar events, mostly stored on internal device memory | [1] Owner, [5] Malicious App (as authorized) | | 2 | Communication Data | Text messages, emails, call logs, mostly stored on internal device memory | [1] Owner, [3] Operator, [5] Malicious App (as authorized) | | 3 | Application data | Custom data stored by browsers, chat, social networking apps, on both internal and memory card; | [1] Owner, [3] Operator (if not HTTP/S or SSL), [5] Malicious App (as authorized) | | 4 | Media files | User generated and download photos, videos and music, primarily stored on memory card | [1] Owner, [5] Malicious App |

## STRIDE Threat List | Type | Examples| | ---- | ---- | | Spoofing | - Detainer[2] or Malicious App[5] may gain control of mobile phone and pretend to be Owner[1] | | Tampering | - Malicious App[5] changes configuration data on the device | | Repudiation | - Malicious App[5] or other system backdoor may disable or block app; - Operator[3] may passively monitor messages and pass the information along to the Detainer[2] | | Information Disclosure | Detainer[2] could have full access to Assets stored on the mobile device; - Detainer[2] may have physical and logical forensic data extraction tools that can override password controls on device and read from "wiped" storage; - Operator[3] may learn identity of Support Org[4] | | Denial of Service | - Communications may be blocked from being sent or received by Operator [3]; - Mobile phone may be disabled by Operator[3] or Malicious App[5] from running remote wipe | | Elevation of Privilege | - Malicious App [5] launches insecured intents or exploits known bug; - Detainer[2] or Operator[3] may be able to impersonate the Owner[1] |

## Security Controls / Mitigation | Type | Tactics | | ---- | ---- | | Authentication (vs. Spoofing) | - Create a a non obvious passphrase for use in app - Lock screen of your mobile phone using passphrase or PIN | | Authorization & Auditing (vs Tampering, Repudiation, Elevation of Priv) | - Do not install any unnecessary, third-party mobile apps with network access; - Scan your mobile device using available malware tools; - Install a firewall or network connection monitoring utility; - Use a non-real name registered SIM card and mobile phone | | Cryptography and Identity Protection (vs Information Disclosure) | - For extra sensitive data, use an app that supports an and password authentication and encrypted database; - Use a mobile OS with disk and memory card encryption; - Use only browser-based HTTPS services that do not store data locally;- Do not store or save web service passwords on your mobile phone | | Alternate Communications (vs Denial of Service) | - Use VPNs or Tor proxying software to hide source IP and traffic; - Use apps/services that work in WIFI only mode if data service disabled; - Use apps that allow device-to-device data sharing |

## SQLCipher Encrypted Database
SQLCipher is an SQLite extension that provides transparent 256-bit AES encryption of database files. It mirrors the standard android.database API. Pages are encrypted before being written to disk and are decrypted when read back.
- SQLCipher has a small footprint and great performance so it's ideal for protecting embedded application databases and is well suited for mobile development. - Blazing fast performance with as little as 5-15% overhead for encryption - 100% of data in the database file is encrypted - Uses good security practices (CBC mode, key derivation) - Zero-configuration and application level cryptography - Algorithms provided by the peer reviewed OpenSSL crypto library.
### CipherKit "Platform" IMAGE
### Defense in Depth Make attacks difficult with multiple layers of security
### Principle of Least Privilege Access to device should not allow access to all apps and data
### Data Security Minimize impact of unauthorized access, on and off device
### Strategies 1. Authentication 1. Encryption 1. Authenticity
## SQLite vs. SQLCipher
``` ~ sjlombardo$ hex dump -C sqlite.db 00000000 53 51 4c 69 74 65 20 66 6f 72 6d 61 74 20 33 00 |SQLite format 3.| … 000003c0 65 74 32 74 32 03 43 52 45 41 54 45 20 54 41 42 | et2t2.CREATE TAB| 000003d0 4c 45 20 74 32 28 61 2c 62 29 24 01 06 17 11 11 |LE t2(a,b)$ …..| … 000007e0 20 74 68 65 20 73 68 6f 77 15 01 03 01 2f 01 6f | the show…./.o| 000007f0 6e 65 20 66 6f 72 20 74 68 65 20 6d 6f 6e 65 79 |ne for the money|
~ $ sqlite3 sqlcipher.db sqlite> PRAGMA KEY='test123′; sqlite> CREATE TABLE t1(a,b); sqlite> INSERT INTO t1(a,b) VALUES ('one for the money', 'two for the show'); sqlite> .quit
~ $ hex dump -C sqlite.db 00000000 84 d1 36 18 eb b5 82 90 c4 70 0d ee 43 cb 61 87 |.?6.?..? p.?C?a.| 00000010 91 42 3c cd 55 24 ab c6 c4 1d c6 67 b4 e3 96 bb |.B?..?| 00000bf0 8e 99 ee 28 23 43 ab a4 97 cd 63 42 8a 8e 7c c6 |..?(#C??.?cB..|?|
~ $ sqlite3 sqlcipher.db sqlite> SELECT * FROM t1; Error: file is encrypted or is not a database
```
https://github.com/sqlcipher/android-database-sqlcipher
``` import net.sqlcipher.database.SQLiteDatabase;
SQLiteDatabase.loadLibs(this);
SQLiteDatabase db = eventsData.getWritableDatabase("my password"); ```
### Simple Steps We've packaged up a very simple SDK for any Android developer to add SQLCipher into their app with the following three steps:
* Add a single sqlcipher.jar and a few .so's to the application libs directory * Update the import path from android.database.sqlite.* to info.guardianproject.database.sqlite.* in any source files that reference it. The original android.database.Cursor can still be used unchanged. * Init the database in onCreate() and pass a variable argument to the open database method with a password*: - SQLiteDatabase.loadLibs(this); //first init the db libraries with the context - SQLiteOpenHelper.getWritableDatabase("thisismysecret"):
### SQLCipher Features - AES 256 CBC - Random IVs - Random salt - Key Derivation - MAC - OpenSSL - Fast startup - No size limit
### How it Works - Pager Codec - Key Derivation - Encryption - MAC
### Advanced - PRAGMA rekey - PRAGMA cipher - PRAGMA kdf_iter - PRAGMA cipher_page_size - PRAGMA cipher_use_hmac - ATTACH - sqlcipher_export()
```

## IOCipher Encrypted Virtual File System
IOCipher provides a virtual encrypted disk for Android apps without requiring the device to be rooted. It uses a clone of the standard java.io API for working with files, so developers already know how to use it. Only password handling, and opening the virtual disk are what stand between the developer and working encrypted file storage. It is based on and SQLCipher. IOCipher is a cousin to SQLCipher-for-Android since it is also based on SQLCipher and uses the same approach of repurposing an API that developers already know well. It is built on top of libsqlfs, a filesystem implemented in SQL that exposes a FUSE API.
### CipherKit “Platform” *image*
### IOCipher: Core Features * Secure transparent app-level virtual encrypted disk * No root required * Only three new methods to learn: new VirtualFileSystem(dbFile), VirtualFileSystem.mount(password), and VirtualFileSystem.unmount() * Supports Android versions 2.1 and above * Licensed under the LGPL v3+
### IOCipher: The Stack * info.guardianproject.iocipher - Java/JNI wrapper API * LibSQLFS / FUSE - Virtual Filesystem that maps to SQL schema / structured database * SQLCipher - Encryption layer for SQLite * SQLite - Base storage mechanism
### Adding IOCipher to App - manage the password - connect to your encrypted disk’s file using new VirtualFileSystem(dbFile) - mount it with a password using VirtualFileSystem.mount(password) - replace the relevant java.io import statements withinfo.guardianproject.iocipher, e.g.: - import info.guardianproject.iocipher.File; - import info.guardianproject.iocipher.FileOutputStream; - import info.guardianproject.iocipher.FileReader; - import info.guardianproject.iocipher.IOCipherFileChannel; - import info.guardianproject.iocipher.VirtualFileSystem; - import java.io.FileNotFoundException; - import java.io.IOException; - import java.io.InputStream; - import java.nio.channels.Channels; - import java.nio.channels.ReadableByteChannel;
### IOCipher Example https://github.com/guardianproject/IOCipherExample
``` import info.guardianproject.iocipher.File; import info.guardianproject.iocipher.FileOutputStream; import info.guardianproject.iocipher.VirtualFileSystem;
File dbFile = getDir("vfs", MODE_PRIVATE).getAbsolutePath() + "/myfiles.db";
vfs = new VirtualFileSystem(dbFile);
// TODO don't use a hard-coded password! prompt for the password vfs.mount("my fake password");
File file = new File(dirPath); File[] files = file.listFiles(); ```
## CacheWord Secure Passphrase Management
CacheWord is an Android library project for passphrase caching and management. It helps app developers securely generate, store, and access secrets derived from a user's passphrase. 1. Secrets Management: how the secret key material for your app is generated, stored, and accessed 1. Passphrase Caching: store the passphrase in memory to avoid constantly prompting the user
### CipherKit “Platform” [Image]
### CacheWord Features CacheWord manages key derivation, verification, persistence, passphrase resetting, and caching secret key material in memory. - Strong key derivation
```

(PBKDF2) - Secure secret storage (AES-256 GCM) - Persistent notification: informs the user the app data is unlocked - Configurable timeout: after a specified time of inactivity the app locks itself - Manual clearing: the user can forcibly lock the application - Uses Android's Keystore on 4.x if available - Not Yet Implemented
### Problem with Android... [IMG]
### CacheWord Solution [IMG]
### CacheWord Code Example
https://github.com/guardianproject/cacheword/tree/master/sample
``` public class CacheWordSampleActivity extends Activity implements ICacheWordSubscriber { … mCacheWord = new CacheWordActivityHandler(this);
@Override public void onCacheWordLocked() {}
@Override public void onCacheWordOpened() { // fetch the encryption key from CacheWordService SecretKey key = ((PassphraseSecrets) mCacheWord.getCachedSecrets()).getSecretKey(); }
@Override public void onCacheWordUninitialized() { mCacheWord.setCachedSecrets(PassphraseSecrets.initializeSecrets( CacheWordSampleActivity.this, “my secret passphrase”)); } ```
## NetCipher Secured Networking
### CipherKit “Platform” [image]
### NetCipher: 3 reasons 1. *Stronger Sockets*: Through support for the right cipher suites, pinning and more, we ensure your encrypted connections are as strong as possible. 1. *Proxied Connection Support*: HTTP and SOCKS proxy connection support for HTTP and HTTP/S traffic through specific configuration of the Apache HTTPClient library 1. *OrbotHelper*: a utility class to support application integration with Orbot: Tor for Android. Check if its installed, running, etc.
### Network Threats Tor Proxying [image]
### NetCipher Code Example https://github.com/guardianproject/NetCipher
``` OrbotHelper oc = new OrbotHelper(this); if (!oc.isOrbotInstalled()) oc.promptToInstall(this); else if (!oc.isOrbotRunning()) oc.requestOrbotStart(this);
StrongHttpsClient httpclient = new StrongHttpsClient(getApplicationContext());
if (pType == null) httpclient.useProxy(false, null, null, -1); else if (pType == Proxy.Type.SOCKS) httpclient.useProxy(true, "SOCKS", proxyHost, proxyPort); else if (pType == Proxy.Type.HTTP) httpclient.useProxy(true, ConnRoutePNames.DEFAULT_PROXY, proxyHost, proxyPort); ```
## From here https://guardianproject.info/contact
Guardian-Dev and SQLCipher mailing lists IRC (freenode): #guardianproject Project Trackers: https://dev.guardianproject.info
support@guardianproject.info
# How do I protect my Data (IOCipher Virtual Encrypted Disks)
Source: Guardian Project
IOCipher provides a virtual encrypted disk for Android apps without requiring the device to be rooted. It uses a clone of the standard java.io API for working with files, so developers already know how to use it. Only password handling, and opening the virtual disk are what stand between the developer and working encrypted file storage. It is based on and SQLCipher.

IOCipher is a cousin to SQLCipher-for-Android since it is also based on SQLCipher and uses the same approach of repurposing an API that developers already know well. It is built on top of libsqlfs, a filesystem implemented in SQL that exposes a FUSE API.
## Features - Secure transparent app-level virtual encrypted disk - No root required - Only three new methods to learn: `VirtualFileSystem.get(), VirtualFileSystem.mount(dbFile, password), and VirtualFileSystem.unmount()` - Supports Android versions 2.1 and above - Licensed under the LGPL v3+
## Adding IOCipher to your App
Here are the things you need to do in your code to make it use IOCipher encrypted storage for all of your app's file storage:
1. manage the password using Cacheword or whatever 1. get the VFS singleton using VirtualFileSystem.get() 1. mount a database file with a password using VirtualFileSystem.mount(dbFile, password) 1. replace the relevant java.io import statements with info.guardianproject.iocipher, e.g.:
For more detailed examples, see IOCipherExample, IOCipherThreadTest, and IOCipherTests. To start from scratch, follow the instructions on starting with SQLCipher-for-Android, then download IOCipher and add it to the libs/ folder of that new project.
## Downloads Here you can get the complete IOCipher jar and native library files, ready to drop right into your project (for MIPS, you need to build from source):
* IOCipher-v0.2.zip * detached gpg signature * MD5: d4fdec3ecfaee96277eaa0372db092ce * SHA1: f89509b802f2982e62b3e5b31cb90266e95262a0
If you are interested in experimenting with the underlying FUSE library, you can download the optional libsqlfs source tarball:
* libsqlfs-1.3.tar.bz2 * detached GPG signature * MD5: 867d60bcd1cb19f09b1fd3b7112767ac * SHA1: 3d5fcd3eef9bf07093987ae98951ab7a4525393a
## Source Code Repositories * all you need for your project: https://github.com/guardianproject/IOCipher
## optional: the test suite: https://github.com/guardianproject/IOCipherTests a simple example file manager app: https://github.com/guardianproject/IOCipherExample a very simple test app: https://github.com/guardianproject/IOCipherThreadTest the core: https://github.com/guardianproject/libsqlfs
## Usage notes * only one active mount per-app is supported * single thread/sequential access is the preferred way of using IOCipher * multi-threaded access possible (potentially unstable under extremely high write load) * VFS now has beginTransaction and completeTransaction to optimize performance * parts of java.io not currently supported: vectored I/O, memory-mapped files
## Reporting Bugs Please report any bugs or issues that you have with this library! We want to hear from you. Help us improve this software by filing bug reports about any problem that you encounter. Feature requests and patches are also welcome!
Report a bug or issue # How do I protect my app's data (SQLCipher: Encrypted Mobile Databases) Source URL: SQLCipher: Encrypted Database
## Introduction In an environment where mobile data privacy is increasingly in the headlines, this project will make it easier than ever for mobile developers to properly secure their local

application data, and in turn better protect the privacy of their users. The data stored by Android apps protected by this type of encryption will be less vulnerable to access by malicious apps, protected in case of device loss or theft, and highly resistant to mobile data forensics tools that are increasingly used to mass copy a mobile device during routine traffic stops.

SQLCipher is an SQLite extension that provides transparent 256-bit AES encryption of database files. To date, it has been open-sourced, sponsored and maintained by Zetetic LLC. In the mobile space, SQLCipher has enjoyed widespread use in Apple's iOS, as well as Nokia / QT for quite some time. Given that Android by default provides integrated support for SQLite databases, our goal was to create an almost identical API for SQLCipher, so that developers of all skill level could use it, without a steep learning curve.

LEARN MORE AND DOWNLOAD SOURCE CODE

## A Simple Example A typical SQLite database in unencrypted, and visually parseable even as encoded text. The following example shows the difference between hexdumps of a standard SQLite db and one implementing SQLCipher.

```
$ hexdump -C sqlite.db 00000000 53 51 4c 69 74 65 20 66 6f 72 6d 61 74 20 33 00 |SQLite format 3.| … 000003c0 65 74 32 74 32 03 43 52 45 41 54 45 20 54 41 42 |et2t2.CREATE TAB| 000003d0 4c 45 20 74 32 28 61 2c 62 29 24 01 06 17 11 11 |LE t2(a,b)$…..| … 000007e0 20 74 68 65 20 73 68 6f 77 15 01 03 01 2f 01 6f | the show…./.o| 000007f0 6e 65 20 66 6f 72 20 74 68 65 20 6d 6f 6e 65 79 |ne for the money|
~ $ sqlite3 sqlcipher.db sqlite> PRAGMA KEY='test123′; sqlite> CREATE TABLE t1(a,b); sqlite> INSERT INTO t1(a,b) VALUES ('one for the money', 'two for the show'); sqlite> .quit
~ $ hexdump -C sqlite.db 00000000 84 d1 36 18 eb b5 82 90 c4 70 0d ee 43 cb 61 87 |.?6.?..?p.?C?a.| 00000010 91 42 3c cd 55 24 ab c6 c4 1d c6 67 b4 e3 96 bb |.B?..?| 00000bf0 8e 99 ee 28 23 43 ab a4 97 cd 63 42 8a 8e 7c c6 |..?(#C??.?cB..|?|
~ $ sqlite3 sqlcipher.db sqlite> SELECT * FROM t1; Error: file is encrypted or is not a database
```

(example courtesy of SQLCipher)

## Details for Developers We've packaged up a very simple SDK for any Android developer to add SQLCipher into their app with the following three steps:

1. Add a single sqlcipher.jar and a few .so's to the application libs directory 1. Update the import path from android.database.sqlite.* to info.guardianproject.database.sqlite.* in any source files that reference it. The original android.database.Cursor can still be used unchanged. 1. Init the database in onCreate() and pass a variable argument to the open database method with a password*: - SQLiteDatabase.loadLibs(this); //first init the db libraries with the context - SQLiteOpenHelper.getWritableDatabase("thisismysecret"):

Note: we are working on some dialog builder helper methods for password and PIN input, password caching, and other features that we would like to standardize across all applications that use SQLCipher.

## Compatibility The Developer Preview implements SQLCipher v1, is compatible with Android 2.2 & 2.3, and works only within one process (you can't pass a Cursor from a remote Service to an Activity).

## Notepad + SQLCipher = Notepadbot Notepadbot is a sample application pulled from the standard Android samples code and updated to use SQLCipher. You can browse the source here and download the apk here.

## Final Notes It’s important to note that this project is not intended to be a distinct, long-term fork of SQLCipher. We’ve been working closely with the SQLCipher team at Zetetic and fully intent to closely maintain the project as SQLCipher evolves, re-integrating changes in upcoming releases such as SQLCipher v2.

The Android support libraries are licensed under Apache 2.0, in line with the Android OS code on which they are based. The SQLCipher code itself is licensed under a BSD-style license from Zetetic LLC. Finally, the original SQLite code itself is in the public domain. # How Do I Protect My App's Data?

intro foo

## Protecting App Databases

## Securing and Hiding Multimedia files and Documents

## Defending Against Network Attacks

CipherKit: We have 3 tools designed for Android app developers to make apps that are able to ensure better encryption and anonymity:

SQLCipher: Encrypted Database SQLCipher is an SQLite extension that provides transparent 256-bit AES encryption of database files. It mirrors the standard android.database API. Pages are encrypted before being written to disk and are decrypted when read back. SQLCipher GitHub Code

IOCipher: Encrypted Virtual Disk IOCipher is a virtual encrypted disk for apps without requiring the device to be rooted. It uses a clone of the standard java.io API for working with files. Just password handling & opening the virtual disk are what stand between developers and fully encrypted file storage. It is based on libsqlfs and SQLCipher. IOCipher Source Code

NetCipher: Encrypted Network Data & Tor Integration NetCipher is improving network security. It provides a strong TLS/SSL verifier to help mitigate weaknesses in the certificate authority system. It eases the implementation of supporting SOCKS and HTTP proxies into applications and also supports onion routing for anonymity and traffic surveillance circumvention.

# How do I secure my internet traffic (NetCipher)

Source: Guardian Project

## Better TLS and Tor App Integration

This is an Android Library Project that provides multiple means to improve network security in mobile applications. The “Onion” name refers to not only the Onion Routing concept used by Tor (which provides anonymity and resistance to traffic surveillance), but also the idea of multiple layers of security that any application should utilize.

More specifically this library provides:

1. Stronger Sockets: Through support for the right cipher suites, pinning and more, we ensure your encrypted connections are as strong as possible. 1. Proxied Connection Support: HTTP and SOCKS proxy connection support for HTTP and HTTP/S traffic through specific configuration of the Apache HTTPClient library 1. OrbotHelper: a utility class to support application integration with Orbot: Tor for Android. Check if its installed, running, etc.

IT MUST BE NOTED, that you can use this library without using Orbot/Tor, but obviously we think using strong TLS/SSL connections over Tor is just about the best thing in the world.

This library was formerly named OnionKit – https://github.com/guardianproject/onionkit

## Stronger Sockets

Developers can create their own CACert store using the information provided by our CACertMan project: https://github.com/guardianproject/cacert
It can be used in combination with Android Pinning and the MemorizingTrustManager, to support user prompted override for non-validating certificates.
## Proxied Connections
Once Orbot connects successfully to the Tor network, it offers two proxy servers running on localhost that applications can route their traffic through.
HTTP Proxy: localhost:8118 SOCKS 4/5 Proxy: localhost:9050
The sample project shows the basics of how to use this library to open sockets and make HTTP requests via the SOCKS and HTTP proxies available from Orbot The standard Apache HTTPClient libraries provide calls to setup proxying. This sample code demonstrates that. All applications using the SOCKS proxy should not resolve their DNS locally, and instead should pass the hostnames through the SOCKS proxy.
## Orbot Helper Provides simple helper to check if Orbot (Tor for Android) is installed, and whether it is currently running or not. Allows your app to request Orbot to start (user is prompted whether to start or not). Finally, it can show a user prompt to install Orbot, either from Google Play, or via direct APK download from torproject.org or the guardianproject.info site. For apps with on-device servers, it can also assists in requesting a Tor Hidden Service from Orbot, and discovering the assigned .ONION address.
# Security and Privacy Chapter
- Encrypt all the bits introduction to mobile security - How to I protect my apps and data (*needs more content*) - How do I encrypt a file (Intro to ioCipher virtual encrypted disease) - How do I protect my app's data (Intro to sqlCipher encrypted mobile databases) - How to I secure my internet traffic (Intro to NetCipher) - Taking smart photos # Taking smart photos
Source: Guardian Project
Obscuracam is a photo and video app for Android that keeps certain information private.
Ever capture someone in a photo or video, then realize they may not want to be in it? Not comfortable posting a friend, family member or child’s face on the internet? Worried about the geolocation data in the picture giving away private hideaway? Tired of Facebook, Google and other sites “auto detecting” faces in your photos? Then this is for you, giving you the power to better protect the identity of those captures in your photos, before you post them online.
Take a picture or load a photo or video from the Gallery, and ObscuraCam will automatically detect faces that you can pixelate, redact (blackout) or protect with funny nose and glasses. You can also invert pixelate, so that only the person you select is visible, and no one in the background can be recognized.
This app will also remove all identifying data stored in photos including GPS location data and phone make & model. You can save the protected photo back to the Gallery, or share it directly to Facebook, Twitter or any other “Share” enabled app.
We’re currently building a version of ObscuraCam for advanced users called InformaCam. Users will have the option to secure images with PGP, inversely, to gather and broadcast additional details in metadata.
## How To Get The Latest Release
You can find the app in the Android Market. For those without access to the Android Market, you can get the ObscuraCam.APK file from our public builds folder. For this option, be sure to

check back for updates, because the app will not auto-update itself.

## Background

We developed our more secure and private camera application in partnership with Witness.org, the leading human rights video advocacy and training organization. This collaboration grew out of an open hackday at the 2010 Open Video Conference.

This is an open-source project and you can track our progress here: https://github.com/guardianproject/SecureSmartCam.

For updates on our progress, you can check out our blog: https://guardianproject.info/tag/obscuracam/

## User Stories Here are some of the user stories we aim to support, along with links where to track our progress and find out more:

1. An activist group records a video interview of a spokesperson at a protest and wants to protect the identify of the people in the background of the shot.

Development of video filtering and processing is underway: https://github.com/guardianproject/SSCVideoProto

1. A rights defender uses their smartphone to take many pictures of abuses in a village. On their way back to their hotel, they are detained by local authorities and their phone is confiscated.

Learn more about our encrypted database library SQLCipher: https://guardianproject.info/code/sqlcipher/ and our Cryptographic features planning: https://github.com/guardianproject/SecureSmartCam/wiki/Cryptographic-features

1. The internet and mobile network are shutdown. Images taken at a protest of human rights abuses need to be delivered out of@gamamb the country. (tor/proxies/bluetooth/wifi mesh)

Xfer is our video upload and download app that works over the Tor network: https://github.com/guardianproject/sscxfer

1. My phone could be compromised. I have already delivered what I need to. I want to remove all traces of any images captured.

InTheClear is our panic/data-wipe app: https://github.com/guardianproject/InTheClear

# Building an Open Source Community

Source: OpenSource.com

I have told the story of how FinTP, the first open source application for processing financial transactions, was born. Here, I would like to present a deeper view on how the community is being built, its structure and governance and why I think people should join such communities. I myself am a founding member of two communities, the first being built together with my friends after our first bachelor party, from a desire to preserve our spirit of joy and good vibes. The second one is the open source community around FinTP, called FINkers United.

An open source project cannot succeed without a powerful community that supports its development. At the same time, Eric Raymond said in The Cathedral and the Bazaar that a necessary pre-condition for success in an open source project is having something runnable and testable to play with. Therefore we have the chicken and egg dilemma of which comes first—the product or the community. Fortunately, FinTP had both, being the result of evolution from its previous commercial version to open source and having a company, partners’ network and a user base that would become the starting point for the open community FINkers United.

While it may seem that the transition from the closed community to FINkers United is destined to be smooth, there are many adjustments to be made. The roles in the community have to change drastically, from the former way of interaction between supply and demand sides, in order to benefit from all the advantages an open environment has to offer—reduced time-to-market, better results delivered from a strong collaboration, cost savings. Of course, a coherent governance strategy has to be established so the project doesn’t go off rails.

## Community structure

Communities are groups of individuals sharing common interests. FINkers United is not different, all members having the same goal to improve the processing of financial transactions based on open source applications, pushing the interoperability level in order to achieve a new semantic standardization for financial transactions. For this specific community and mission, several professional profiles of individuals and institutions are welcomed: technical (developers, IT architects and designers, implementers, support, quality assurance), business (business knowledge in banking practice, standards, financial transaction processing, banking or corporate treasury, corporate to bank business etc.), communication, marketing, market analysts, auditing and legal consultants.

In order to successfully establish the strategy and objectives, to manage and control the activity of the community, the community we conceived has the following three governing entities: General Assembly, Board of Directors and Censor. In the early stages of the community, the governance is ensured by the founding company, since it is the sole contributor. For now, the community has a Technical Committee coordinated by the company’s CTO, which includes 10 project leaders. In time, new hierarchies will emerge based on merit and contributions, so any member can be elected.

## Reasons to join the FINkers United community

The most common question is what’s in it for me, *why should I join this community?*

The rewards are very different from the traditional closed-source projects, where there is a clear limit. For suppliers, the types of reward-earning contributions are in the job description, for which you get a monthly paycheck. For the demand side, what you pay is what you get, with limited flexibility from both providers and budget owners.

In open source communities, the benefits are different for every member profile, but the essence is that everybody is involved because of having a shared interest and acts as an entrepreneur—having ideas, initiatives and the freedom to act accordingly to own beliefs.

### For the *adopters*, the benefits of using and supporting open source software have been debated in length, and they include:

- The power to influence projects and strategies - Freedom from vendor locking - Benefit from better, more secure and reliable products - Better TCO and time to market - Reduce development and maintenance cost - Attract and retain development talent - Community based, price affordable, service consistent, guaranteed quality 'escrow agent'

### For *contributors*, the satisfactions are mostly triggered by a sense of accomplishment.

- **Material**—custom professional services can be offered to adopters who lack certain skills or need SLAs - **Intellectual**—there’s great opportunity to develop the professional profile by taking on the technical challenges open source projects have to offer - **Recognition**—open source communities reward each of their members based on the value of their contributions - **Competitive**—the opportunity to work for state of the art technological projects, with high impact on several industries

Speaking for myself, my motivation for joining FINkers United is that being a part of the team

that attempts to revolutionize the financial transactions processing arena is a great challenge and experience, not to mention that succeeding would be truly an accomplishment. Moreover, I believe that creativity is original thinking based on openly sharing knowledge and transparently sharing results.
Why would you join an open source community? # Five ways to boost community engagement
Source: OpenSource.com
The room was packed with almost 30 community managers from around the world. We were attending the Community Leadership Summit in Portland, Oregon and I was leading a session called *Building programs and other ways to engage your community*. During the session, I shared a few of the programs that we've implemented here on Opensource.com to foster community engagement. But the session wasn't just about our community, it was an opportunity to hear from other community managers in open source out there in the trenches.
We had community managers from Mozilla, Open Source Institute, Puppet Labs, Chef, and other organizations. We talked about what's working well and what's not. What's been tried and tested. And, who has new ideas? Together we came up with a list of techniques that can be used to keep our open source communities engaged.
1. **Welcome new community members**
Whether it's in-person or virtually, make sure new community members get a warm welcome. For in-person events, consider having a designated greeter, preferably someone outgoing who wears a smile and is knowledgeable about your group or organization.
For a virtual welcome, an email greeting should work just fine. The more personal, the better. Never automate it. And it should be from your community manager or someone who might interact with the community at a later time.
1. **Conduct a value assessment to determine interest or expertise**
One community manager at the conference explained how they perform a value assessment to understand why someone has joined their community and how they want to participate. They do this about a week after the welcome email goes out and use the data to determine the next steps to take for community engagement.
A technique like this will not only help people find their "why," or place within the community, but it can also help community managers understand what they can do to help the community member thrive in the group.
1. **Sponsor community members to attend events**
Many organizations will cover expenses for people to attend events. This allows them to have in-person meetings and face-to-face time with other community members. The relationships formed from being at in-person at meetings, conferences, and other events can give a real boost to community members.
1. **Break down the silos**
Another community manager at the conference shared a technique they use within their organization to help break down barriers to entry to a community or engagement within that community. On a monthly basis, this community manager organizes an online meetup to promote community engagement. They ask members what they want to learn about, help them find a source matter expert, then host a meet-up on the topic. The community gets extreme value out of these events, and their overall community engagement has increased.
While this example was referring to breaking down silos within a large corporate organization, I

think the idea has merit and can be applied to groups of all kinds .
1. **Say thank you**
In many communities, the words "thank you" are not used enough! Say thank you, and then thank that person or group again. In our session, we came up with a list of several ways that you can thank your community as well as individual members.
- Send them stickers. - Give them a shout-out in your newsletter or blog, on social media, etc. - Tag people and post pictures in social media (be sure to get permission first). - Send them T-shirts, hats, or other swag. - Feature them in a “community spotlight” in newsletters or on your website homepage. - Thank every contributor who contributed code or content at certain milestones (perhaps during project releases). - Send them a handwritten letter. - Celebrate with your community or support a way for them to celebrate on their own. - Provide them sponsorship to attend an event.
While the session only lasted an hour, the ideas we shared during it will live on and continue to evolve as our communities change and grow. Community managers and others can join the conversation in the Community Leadership Summit online forum. And as always, I welcome you to share some of your experiences in the comments. # How to use IRC (Internet Relay Chat)
Source: Instructable
[EDIT] This is an Instructable intended as a starter for those who do not yet understand Internet Relay Chat, or IRC. This project is not intended to cover the entire scope of IRC and each individual client's capabilities, but is targeted to the windows user who is new to IRC, and this project aims to help those users to get started with IRC as a superior form of communication over IM clients such as AIM or YIM, which require installing software already proven to be a security-risk. Even if you do not choose to use Chatzilla or Firefox, some information here may be useful to new IRC users with general commands supported by most IRC clients. If you are looking for help with a specific client, see FAQ's related to your client specifically.
If you have more wisdom to add, please post it as a new comment. For those new to IRC, please read this project and it's comments anyway to make sure you know the basics. You are a user until made an operator, so don't try to kick ops from public channels or your client will laugh at you. All IRC addresses and/or examples are valid and accessible for the purpose of learning how to use the system, to the best of my knowledge. Screenshots were made from actual logons and were not "doctored" in any way, so you can join any channels, public or hypothetical, displayed in any screenshot here.
[/EDIT]
Internet Relay Chat is much like your instant messengers, but is devoid of spam or general security risks that other IM services not only allow but are built to accept. Typical IM services such as YIM, MSN-chat, AIM, and many others are specific targets due to their accessibility and ability to silently upload files to your computer and execute them on command without you even knowing what happened, from someone not even showing up in your chat window.
IRC is not only secure in multiple ways, but you cannot hide behind an application. If someone's there, you'll know it, and you must accept all file transfers before they are actually received. Below, I'll explain how using IRC is just as easy, far safer, and much more functional than your bloated n00b IM-client-for-dummies. For those smart enough to use Firefox, this can't be easier. For those still using IE for anything, there may be no saving you, but I'll try. Step 1:

Learn to use Firefox, or download a suitable IRC client such as mIRC
## Step 1: Learn to use Firefox, or download a suitable IRC client such as mIRC
* You can find Firefox here * Get the Chatzilla addon here * For those so stubborn not to try the best, you can get mIRC here
I'll focus on those who know better to use Firefox than IE. Downloading the Chatzilla extension and incorporating it into Firefox is easy, simply download it and it knows where to go. You may be asked to restart Firefox or your computer, if you are not asked, restart your computer just to assure a correct install anyway. Windows likes to corrupt itself and blame properly-coded software.
Now that you have it installed, you can find chatzilla from the top by dropping down the "tools" tab in the menu-bar. Step 2: Join a network
## Step 2: Join a network This seems largely difficult, but it is incredibly easy, and your choices are laid out for you. In this image, I have joined "efnet" as my primary host and server. By now you should have connected to efnet for sake of example.
IRC addresses work similarly to http "hyperlinks" (aka "links"), so that clicking on a link to an irc channel will automatically start chatzilla for you. Below, I will get you started.
Start chatzilla, and you will be faced with a lot of messages. Ignore them for now and type the following into the text-box at the bottom:
/networks
Chatzilla will return this to you in the message window (or similar):
{INFO} Available networks are dalnet, efnet, freenode, hispano, hybrid net, ircnet, moznet, quake net, serenia, slashnet, undernet, webbnet.
The networks will be like hypoerlinks, so you can just click on one to join that particular server. Here I have chosen to use "efnet" as my server by simply clicking on the name shown above. Now that I am on a network, I have the ability to join an existing "room" (the real name is a "channel", so we are calling it a "channel" from now on), or create a new one (it is best to start your own channel than to hijack an existing one). So, let's say you and your friends decide to meet on irc://efnet/instructables. Firefox users can simply type this address into the URL window in Firefox, and chatzilla will automatically connect to that address. If you have it installed, try it below by simply clicking on the link:
[irc://efnet/instructables irc://efnet/Instructables]
Congratulations, you are now using IRC. The link above is shown just as you would type it into Firefox's URL textbox (the box at the top that shows your current internet address). Remember to type an irc address as shown above. Generally, "http" means "internet", "irc" means "irc", of course, and "file" means a location on your computer. You may start to see how the internet and your computer work much in the same way.
Now, say you don't want to use irc://efnet/Instructables, maybe you want your make your own channel to chat in, and you want to call it "Jedi Knights". You can't make this work properly because "channels" (what some call "chat-rooms") cannot have spaces in the names. The ideal workaround to this is to use an underscore instead of a space, as seen below (go ahead and try it):
[irc://efnet/Jedi_Knights irc://efnet/Jedi_Knights]
Notice that I still have efnet in the address. This is because efnet is the server that is hosting your traffic (your text, files, etc). By now you might start to see how you can create your own

places to chat. You need to use a valid server, but you can make up any name you want for your channel as long as it contains only regular characters such as letters and numbers, no special characters will be recognized. This helps your security because an underscore can be printed by over a hundred different key combinations (underscore is often used to display a "non-display character").

You don't even have to use efnet, just type /networks to see what networks are available, and your address needs to be prefixed with a valid IRC server as seen when typing the /networks command. You cannot simply create your own server, and doing so will not be covered here..

Click on the name and you can find or create your own channels (or "rooms"). What server you use amongst those available does not particularly matter except for their performance, and they all are generally reliable. However, you cannot go to irc://efnet/instructables, and be in the same place as irc://dalnet/instructables because you will be on a different server.

## Step 3: Now that you are on IRC...

Picture of Now that you are on IRC... With IRC, you tell the "client" (chatzilla, in this case) that you are giving it a command. The standard is to precede the command with a slash. As seen previously, typing /networks tells the client to display the available networks. To see a list of all available commands, simply type /commands. A list will be displayed showing the proper commands that are accepted from where you are. I won't get into them all, but I'll show you the important and most useful ones.

Probably the most useful to anyone at first is the /nick command. This changes your "screen name" to whatever you want, within a limit of 9 characters. Let's say you started with the nickname "IRCMonkey" as chatzilla defaults to, and you want to be "B_Kenobi". Simply type as shown below:

```
/nick B_Kenobi
```

Now that is your new nickname. By default, chatzilla will save this as your default nickname for that channel with a limit of 9 characters. When you select "open this channel at startup" (selectable as shown in the image displayed below, this becomes a new default startup setting), you simply have to start chatzilla and your username and preferred channel will automatically start and you are ready to go. You will not have to change your nickname every time you start, as chatzilla will retain your last nickname before you close it to be used when you return to that channel.

If you are stuck, or just want more information, simply type /help and chatzilla will walk you through any command. For example, we will ask chatzilla how to use the /dcc-send command:

```
/help dcc-send
[USAGE] dcc-send [ []]
[HELP] Offers a file to |nickname|. On a query view, |nickname| may be omitted to send the offer to the query view's user. A file may be specified directly by passing |file| or, if omitted, selected from a browse dialog.
```

(Please note that the above apparent links do not lead to anything and were not intended)

This may seem confusing, so let me explain it. Say you are chatting with your friend "Gozer", and you want o send them a file called "my_pix.bmp" from your desktop. You simply type:

```
/dcc-send gozer my_pix.bmp
```

Gozer will receive an offer to accept or deny this file, and if they choose to accept it, a new tab will open on your side showing the progress of the file transfer and the rate at which it is being

transferred. The file transfer is directly from your computer to theirs, and no other people chatting with you will even know it is taking place, much less the server itself. It is transferred by a process called "P2P", which is a direct connection directly between you and your recipient. It is not encoded, but the security lies in that the transfer cannot be redirected, located, or even observed because it is just between you. The two computers are passing data as if they were connected by a private network, and cannot be intercepted because only two connections will work....The sender and the receiver. There is virtually no way a third party can get on that data-stream because the connection will either block them or self-terminate, as it is not compatible with more than two users. The transfer will continue in the background as you continue chatting, so there is no need to wait for it to finish.
There is a variant of the /dcc protocol, which is /dcc-chat. This allows you to bypass the server and the two of you act as your own. in a /dcc-chat enabled conversation, you are literally talking just between your two machines on a direct line. Locating this type of chat protocol is nearly impossible unless someone is specifically monitoring your line. to enable this with your friend Gozer, just type:

```
/dcc-chat gozer
```

A new tab will open, and you two are literally chatting client-to-client, without a server as a middle-man. This is rarely necessary, but if you are that paranoid, it is available. You can take a file-transfer automatically from Gozer by typing:

```
/dcc-accept-list-add gozer
```

Now you will automatically accept files from Gozer after a 15-second delay in which you can choose to deny the transfer. If you take no action, the transfer will start and you will begin downloading the file from their computer. There is no limit to file size whatsoever when using DCC. Send 1kB, or 190 GB, size does not matter, so long as you remain connected to Gozer (or whoever your friend is, we're using "Gozer" as an example). Now you will no longer have to choose "accept" every time they send you a file, and there is never any limit to how many transfers you can take at once, however, you will be taking files into the default folder. To change this to your desktop (for ease of explanation), have Chatzilla open and use the toolbar. Go to Chatzilla->Preferences....Go to "global settings" and select the DCC tab..Under "Downloads Folder", type-in "file:///C:/WINDOWS/Desktop/" (without quotes) or browse the path to your desktop and all DCC transfers will go there from now on as long as you "apply" the change before closing the menu.
For reliability, it is best to take one at a time though. This is active as long as they keep the same username, usually, even if their IP changes, such as if they use Dail-up/telephone internet access (The above "IP" and Dial-Up..." are intended links, so clicking on them will give an explanation if you do not know what they mean)
You can PM other users by typing:

```
/msg Gozer I think he's lying
```

This will send a message only to Gozer: "I think he's lying" (without quotes). No other person will ever see this message, and it will show up in a new tab on the chatzilla client only for the recipient of the message, often with a system-sound to alert the recipient. This is a way to talk privately during a chat with several people and only talk to one person, like whispering to them, but noone can ever know that you did or what you said except who you said it to. Experiment with it to see how it works.

The context is /msg , just be sure to add a space between the username you want to talk to and the message you intend to type.
## Step 4: Operator status and it's priveleges...
When you are the first user to show up to a channel, you are automatically given "operator status", meaning you are the operator of that channel, or the administrator of that channel. This gives you special privileges that other users cannot use without that status. It is considered polite to give operator status to people you expect to be there with you, so that they can have administrative control of the channel as well. To share operator status with your friend Gozer, type the following (assuming you were there first):
/op gozer
This command can only be given by someone with operator status to share the same status with a fellow user. What good is "operator status" to the two of you? I'll show you more commands that will demonstrate:
/kick Mr_Rude
This will kick the intrusive person named "Mr_Rude" from the channel, removing them from the channel. A message can be appended to show a reason why you kicked them, but is not necessary. An example is shown below:
/kick Mr_Rude No swearing is allowed on this channel
When Mr_Rude is kicked they will receive the message, "You have been booted from #instructables by B_Kenobi (No swearing is allowed on this channel)"
If you have a persistent user who annoys you and does not respond to the /kick command, you can use the /kick-ban command, that will kick the user off and ban them from rejoining. You can also attach a message in the same way as a simple /kick:
/kick-ban Mr_Rude No swearing is allowed on this channel
They will see the same message as above, but "booted" will be replaced with "banned", and they will not be able to join your channel as long as you are on it. you can remove this by typing the following:
/unban Mr_Rude
Now the ban is lifted and that user can log into your channel again. If another operator banned a user, you can unban them and override their ban as a fellow operator.
Operators can also set the topic for the conversation. To do so, simply type /topic and enter what you want as your topic, to be shown for all users at the top of the screen. Anyone who joins will see this in the introduction to the channel, and all users already logged-on will see the change as it is made:
/topic I love bunnies!
The topic will change to this after this command is given, if you have operator status. Try it and see by looking at the top of the window...
If you give someone operator status and they misbehave, an operator can strip another operator of his status by using the following command:
/deop Gozer
This will remove Gozer's operator status (maybe because he bans your other friends for no reason or abuses his operator status), and remove his ability to change topics, or kick/ban other users. If you wish to not have operator status, you can deop yourself as well, as an operator has control over privileges of all users, including themselves. An amusing way to leave as an

operator is to kick yourself. Kick yourself for foul language even when you haven't used any, or for any reason. It's an inside-joke amongst veteran IRC users.
It takes practice, but not long before you are perfect with these commands....
## Step 5: More commands and interesting fun
Picture of More commands and interesting fun More fun you can have with IRC is using colors and bold when you type, to emphasise what you are saying, and how you say it. Give your text personality by the commands below:
By prefixing /me to a statement, you display your text in italics like a footnote or a silent action by context. an example is below:
Gozer: I really like the new Heidi Klum pix
(you type the following in response):
/me hears that
The result is your post showing as:
B_Kenobi: hears that (in italics)
as if you said it under your breath, or otherwise meant it as a silent thought. Experimentation with this will show how this can be useful in context.
Some of you may wish to type in color or otherwise customise your text. To do so, you need to prefix the text with the appropriate symbol. Start the desired text with the percentage symbol "%" and a small tool-tip will appear showing the supported commands. For example, %U means to underline the following text. you can end the effect by repeating the command. For example:
I %Uwant money %U man!
The result is "I want money man", but only "want money" is underlined in that text. Bold is a lot simpler:
I *want money* man
As typed, "*want money*" will show in bold text. Anything bracketed in asterisks will be shown as bold type, but the asterisks will show. Using asterisks are shorthand, but for more perfect text, use the %B tag instead so that they won't show. Remember that to use this option, you must use capital letters for every prefix and suffix (not %b, but %B for bold).
You can combine tags such as the following:
I %U %B want money %U %B man
Which will show "want money" in bold and underline at the same time. Skillful tagging can show the same in bold/underline//italic all at once. Below I show basic text tagging and it's effect. The commands are as follows (try them when on IRC to see the effect):
I %Bwant money%B man
I *want money* man
I %Uwant money%U man
I %B %Uwant money%B %U man
The image shows the difference between these commands. Many commands can be combined for the desired effect. Note that you do not need to space between tags and the text as a "text tag" (called a "switch") for the client to interpret everything following as the prescribed format until terminated, as seen above. A switch (%U or %B is called a "switch", just like how you turn the lights on in a room) is set in text, and then terminated to return to the default for following text. "%" is a switch, just as "/" is a "command flag". The client responds to "/" as a command and "%" as a switch, and this is how you can better control the client. A switch is a trigger for a

different interpretation of the following text, just as "/" tells the client that the following text is expected to be a command.

I %C9 %U %BWant Money%C9 %U %B

This will show the latest example in bold, underline, and in green. Spaces between the text and the switches are displayed, so if you spaced the tags here, the space would carry the same underline. I keep switches in order, but as long as you close them all, the effect is turned-off. The order is not specific, just as long as each switch is closed as it was opened, or it remains until closed. Text entered into IRC below produce the same result:

I %C9 %U %BWant Money%C9 %U %B I %C9 %U %BWant Money%U %B %C9

This line has all the same switches, and upon conclusion of the line, all switches were opened, and then closed, returning the text back to it's default state. If you omit the %B switch, all your text will be in bold until you "turn it off" on that line..Clients will vary, but chatzilla tends to return to default for the next line. Type the same line above and then add text beyond that. Now try removing a switch, such as "%U"....The effects will continue, but without the underline effect What you see is what you are sending, and another chatzilla client will show the same thing to who you are sending to.

By now, you should be learning something, and are becoming more proficient in computing, even if you are not aware of it. You can get really good with IRC and show-off your skills once you learn how to control it, and some of this should help you pick-up on these tricks.

Now you should be able to use Chatzilla with some proficiency, as well as many other IRC clients. Stop using IE, and start using Firefox, and get the chatzilla extension (no, it's not even on the level of "difficulty", it's child's-play, so you have no excuse), and when you accept the best, you will be above the rest and learn IRC to discover the real IM client that has been around long before YIM or MSN or any other. Use the best, and never need the rest....

All you need to do now is to convert your friends, and show them that you know the real thing from the fake. You don't need to install some unstable application just to connect to this network, you just use the best browser for your internet and your chat-system is right there.

If you still think Internet-Explorer is better than Firefox, and refuse to change, forget this Instructables project, and invest heavily in anti-virus programs and everything else you can, and be sure to have a technician on-call to solve your IE-related problems because you choose to be incorrigible. I'll try to feel sorry for you that your 3.1GHz machine is slower than my 1.3GHz machine due to your gaping security-holes that make your machine a zombie for others to enslave, but you choose that by choosing IE, so enjoy the finest of Microsoft programming and the privileges of your inability to accept change despite the evidence contrary to your popular belief.

If you are conservative in software, you automatically fail, so take pride in your failure of self-designed obsolescence and your inability to accept something above the status-quo, because IRC is simply not for you. IRC requires a minimal level of intelligence, so if you think you are going to do this from Internet explorer, just forget it, this project is light-years from your capability or potential skill level. Continue to use IM until you "are all growed-up" and can handle IRC.

The rest of you are already chatting on IRC now, so enjoy, and welcome to the high-tech world in high-tech format. Never use a dumbed-down client again, because now you have real control, and noone has to hold your hand or distract you from what's going on behind your back.

Welcome to the most secure IM system that was always available to you but noone ever told you how. Now you can chat securely without spam or intrusion, and you have easy and significant control over your chatting area.
Welcome to the 21st century of IM chatting...Enjoy... # Team and Project Management
- Building an open source community - How to use IRC (Internet Relay Chat) - The story of self and motivating a movement - TRAINERS NOTES: STORY OF SELF - Top five open source project management tools - Five ways to boost community engagement # STORY OF SELF
Source: NOI's Story of Self
## The story of self and motivating a team
How can you lead others if they don't know who you are, where you come from, and what your values are? Mastering the art of public narrative allows you to establish firm ground with your constituency in which to collaborate, lead and find common purpose. Each of us has a story to tell that can move others to action. Story of Self is the first part to public narrative where you will learn to convey why you are called to leadership in your movement.
This is from New Organizing Institute
“Why am I called to lead?”
## ACKNOWLEDGEMENTS
We welcome your suggestions for improving this guide further for future trainings. We also welcome you to use it and adapt it for your own trainings, subject to the restrictions below.This workshop guide has been developed over the course of man trainings by Liz Pallatto, Joy Cushman, Jake Waxman, Devon Anderson, Rachel Anderson, Adam Yalowitz, Kate Hilton, Lenore Palladino, New Organizing Institute staff, MoveOn Organizers, Center for Community Change staff, Jose Luis Morantes, Carlos Saavedra, Sean Thomas-Breitfeld, ShuyaOhno, Petra Falcon, Michele Rudy, Hope Wood, Kristen Dore and many others.
## RESTRICTIONS OF USE

## public narrative: story of self
OBJECTIVES: By the end of this training, you will…
- Learn the basics of how public narrative works: values, emotion & story structure - Learn

criteria for an effective story of self and coach others on improving their storytelling - Practice and get feedback on your own story of self
### EACH OF US HAS A COMPELLING STORY TO TELL
Each of us has a story that can move others to action. As you learn this skill, you will be learning to tell a story about yourself, the community you organize with, and your strategy that motivates others to join you in creating change. In addition, you will gain practice in listening, and coaching others to tell a good story.
### PUBLIC NARRATIVE IS A PRACTICE OF LEADERSHIP
Leadership is about accepting responsibility for enabling others to achieve shared purpose in the face of uncertainty. Narrative is how we learn to make choices and construct our identities and purpose—as individuals, as communities and organizations, and as nations.
What does public narrative have to do with this definition of leadership? You can’t ask others to follow you if they don’t understand what your intentions are, and why you are called to lead
### THE HEAD & THE HEART
There are two ways we understand the world: through our head (strategy &analysis) and through our heart (story & motivation). To enable others to achieve shared purpose, public leaders must employ BOTH the head and the heart of their constituency in order to mobilize others to act on behalf of shared values. In other words, they engage people in interpreting *why* they should change their world (their motivation) and *how*they can act to change it (their strategy). Public narrative is the “why”—the art of translating values into action through stories.
### VALUES INSPIRE ACTION THROUGH EMOTION
We don’t think our values; we feel our values. Often we don’t realize what we value in the world until we hear a story or witness an injustice that stirs emotions within us. Emotions inform us of what we value in ourselves, in others, and in the world, and they enable us to express the motivational content of our values to others. Because stories allow us to express our values not as abstract principles, but as lived experience, they have the power to move others to action.
#### Exploring our values:
Each photo below reveals a human story connected to an injustice. Review the photos below to further explore your own values and the process in which you come to realize what you value. Pay attention to your feelings--what specific emotions do you feel? Did you realize you value something that you may not have articulated before? Do you feel inspired to take action?
| | What feelings or emotions does this photo evoke? | What value does this reveal that you hold? | | ---- | ---- | ---- | | HURRICANE KATRINA | | | | ICE RAIDS | | | | OIL SPILL | | | | THE ECONOMY | | |
### SOME EMOTIONS INHIBIT ACTION, OTHERS MOTIVATE ACTION
Public leaders often encounter individuals or groups where mindful action is inhibited by inertia, fear, self-doubt, isolation, and apathy. The job of a leader is not to tell people to stop feeling this way but rather use storytelling to move people from feelings of stagnation to feelings of motivation - urgency, hope, YCMAD (you can make a difference), solidarity, and anger. The language of emotion is the language of movement—they actually share the same root word. Stories mobilize emotions of action to overcome emotions that inhibit us from mindful action.
### PUBLIC NARRATIVE COMBINES A STORY OF SELF, US AND NOW

#### STORY OF SELF
By telling a “story of self” you can communicate the values that move you to lead. Public leaders face the challenge of enabling others to “get” the values that move them to lead. Effective communication of motivating values can establish grounds for trust, empathy, and understanding. In its absence, people will infer our motivations, often in ways that can be very counterproductive. Telling our story of self can help establish firm ground for leadership, collaboration and discovering common purpose.
Every one of us has a compelling story of self to tell. We all have people in our lives (parents, grandparents, teachers, friends, colleagues) or characters we love, whosestories influence our own values. And we all have made choices in response to our own challenges that shape our life’s path— confrontations with pain, moments of hope, calls to action.
The key focus is on our choices, those moments in our lives when our values moved us to act in the face of challenge. When did you first care about being heard? When did you feel you had to act? Why did you feel you could act? What were the circumstances, the place, the colors, sounds? What did it look like? The power in your story of self is to reveal something of those moments that were deeply meaningful to you in shaping your life—not your deepest private secrets, but the events that shaped your public life. Learning to tell a good story of self demands the_courage of introspection_, and of sharing some of what you find.
#### STORY OF US
By telling a “story of us” you can communicate values that can inspire others to act together by identifying with each other, not only with you.
Just as with a story of self, key choice points in the life of a community—its founding, crises it has faced, or other events that everyone remembers—are moments that express the values shared. Consider stories that members of your group have shared, especially those that held similar meaning for all of you. The key is to focus on telling a specific story about specific people at a specific time that can remind everyone – or call to everyone’s attention – values that you share. Telling a good story of us requires the_courage of empathy_ – to consider the experience of others deeply enough to take a chance at articulating that experience.
#### STORY OF NOW
By telling a “story of now” you can communicate an urgent challenge we are called upon to face, the hope that we can face it and the hopeful outcome we can create together, and the choice we must make to act now.
A story of now requires telling stories that bring the urgency of the challenge alive: urgency because of a need for change that cannot be denied, urgency because of a moment of opportunity that may not return. A story of now also offers hope—not make believe hope, but real, plausible hope, often grounded in what others are already achieving, grounded in the courage of others’ actions, and in the strategic vision of what we can achieve together. At the intersection of the urgency and the promise of hope is a choice that must be made – to act, or not to act, to act in this way, or in that. Telling a good story of now requires the_courage of imagination_, or as Walter Brueggemann named it, a prophetic imagination, in which you call attention both to the pain of the world and also to the possibility for a better future.
### STORY STRUCTURE: CHALLENGE, CHOICE, OUTCOME
Every human story has a plot. A plot begins with a challenge that confronts a character with an urgent need to pay attention, to make a choice for which s/he is unprepared. The choice yields

an outcome, and the outcome teaches a moral.
A good story allows the listener to empathetically identify with the character and“feel” the moral. We hear “about” someone’s courage; we are also inspired by it.
The story of the character and his or her choices encourages listeners to think about their own values and challenges, and inspires them with new ways of thinking about how to make choices in their own lives.
#### Incorporating Challenge, Choice, and Outcome in Your Own Story
There are some key questions you need to answer as you consider the choices you have made in your life and the path you have taken that brought you to this point in time as a leader. Once you identify the specific *relevant* choice, dig deeper by answering the following questions.
Challenge: Why did you feel it was a challenge? What was so challenging about it? Why was it your challenge?
Choice: Why did you make the choice you did? Where did you get the courage (or not)? Where did you get the hope (or not)? Did your parents or grandparents’ life stories teach you in any way how to act in that moment? How did it feel?
Outcome: How did the outcome feel? Why did it feel that way? What did it teach you? What do you want to teach us? How do you want us to feel?
A word about challenge. Sometimes people see the word challenge and think it means describing the worst misfortunes of our lives. Sometimes those are the moments that most shaped us. But keep in mind that a struggle might also be one of your own choosing – a high mountain you decided to climb as much as a valley you managed to climb out of. Many things may have been a challenge to you and can be the source of a good story to inspire others.

VIDEO REVIEW

SAMPLE STORY OF SELF, US, AND NOW

We'll be watching a sample story of Self, Us and Now. While you watch it, think about the elements of SELF – US – NOW that you hear in this story.

SELF	**US**	**NOW**
1. What experiences and values call this person to leadership? **2.** What choice points does the speaker include to show, rather than tell us his or her values?	**1.** Who is the “us” that the speaker identifies? **2.** What are the common values the speaker appeals to? How? **3.** What challenges and hopes does this “us” or community share?	**1.** What urgent challenge does this speaker identify? **2.** How does he or she make that challenge real? **3.** What gives us real hope that we can do something? **4.** What is the first step that each person can take to be part of the solution?

1. **What was the speaker’s purpose in telling these stories? What was s/he moving people to do?**
2. **What values did this story convey? How? By telling or showing?**
3. **What details or images in particular reflected those values?**
4. **What were the challenges, choices and outcomes in each part of his story? What morals do the outcomes teach?**

## SAMPLE STORY OF SELF: Lilian Molina

>As told at Powershift 2011: a gathering of 10,000 climate activists from around the US.
Greetings, My name is Lilian Maria Molina and I am the Environmental Justice Director at Energy Action Coalition. I am part Mayas-Chorti, Lenca and Palestinian, was born in Honduras, Central America and moved to the United States at the age of 5 with my mother. For the first couple years my mom and I would take an hour-long ride on a two-floor train; I would always rush to the top floor, look out the window, and envision what I would do at our destination. I would imagine the cartoons I would watch, salivate over the Kudos and Pringles I would be able to eat, and think about all the great toys I would play with. Then one day, as I was playing with a fully equipped Barbie Mansion, my mom reached over and handed me a bottle of Windex and paper towels; at that moment I realized that our hour-long train ride wasn't a field trip, it was a commute to work. My mom and I were there to clean houses not to play.
From that moment on I started to notice that things looked very different in different parts of town. I wondered why some families lived in three floor homes, while I lived in a one-bedroom basement apartment with two families. I wondered why the park equipment in my neighborhood was always broken, but was fancy and new on the other side of town. I wondered if people in the neighborhood where my mom and I cleaned houses had to worry about La Migra coming to their jobs or their homes. I wondered if the kids at these houses ever had to miss school to translate for their parents. I wondered why the police didn't arrest kids around these houses for standing on the corner but my friends back in the neighborhood were arrested all the time. I slowly started to understand that these were two separate worlds.
As I got older, I would refuse to take the hour-long train ride with my mom, instead I would hang out with my friends in the neighborhood. When I was 12, my mom noticed that I was starting to get involved in some risky activities. She decided to send me to Honduras for the summer to spend time with Mi Abuelita (grandma). That summer Mi Abuelita, a Natural Healer and Master Gardener, helped me connect to my ancestral roots and taught me how to love nature through gardening. I learned about all the different plants that she used to help heal people and deliver babies - it was an eye-opening experience. That summer I also realized that some of the people that looked like my family and I wore suits to work and lived in houses rather than apartments.
When I came back to the U.S, I returned to hanging out with my friends; but when I was 16, I decided I was done watching my friends get beat up, get beat by the cops, or arrested. My friends and I started hosting different activities to keep our friends from joining street gangs. Throughout high school we organized different events, from parties, to walkouts to bring awareness to the violence in our communities. Around this time I remembered how the garden that Mi Abuelita introduced me to helped me to heal, and started wondering if a garden in our community could have the same impact for other young people. I got super excited and started looking for plots of land around the school. But in my search I learned that most of the land in Little Village was contaminated with industrial pollution. I thought to myself, "You have to be kidding me, on top of all of the issues I was aware of, our land is also polluted? We have poor education, gang violence, police brutality, immigration raids, militarization of schools and we also have contamination in our community? What the heck else could be wrong?" I learned that what my community was experiencing is called Environmental Racism and what we need is Environmental Justice before we can plant gardens here in Little Village…and that is what brought me to the work that I am doing now.

Now I am here at Power Shift with Front-line Community Members and our Allies, working with the leadership of front-line communities and helping them create a trans-local movement to oppose corporate power is where there is strategic need for youth leadership.

TEAM BREAKOUT SESSION

GROUP PRACTICE WORK

GOALS - Practice telling your Story of Self and get constructive feedback - Learn to draw out and coach the stories of others

AGENDA

Total time: 65 min

1.	**Gather in your team. Choose a timekeeper. Do quick introductions (name and hometown). Articulate group agreements for how you'll work together as during this training. Have your facilitator tell their 2-minute story of self as an example.**	**10 min.**
2.	Take some time as individuals to **silently develop your "Story of Self**." Use the worksheet that follows.	10 min.
3.	**Choose a partner Practice telling your story of self.** - 2 minutes each to tell your story - Focus on the values you want to convey—what specific experiences shaped those values in your life? - Be specific and give lots of detail **Use the worksheet " Coaching Tips: Story of Self" to help guide your feedback**. - 3 minutes each for feedback: - What **values** did the storyteller convey? How specifically? - What is the **Challenge**, **Choice**, **Outcome** in each story? - Were there sections of the story that had especially **good details or**	15 min

1. **Gather in your team. Choose a timekeeper. Do quick introductions (name and hometown). Articulate group agreements for how you'll work together as during this training. Have your facilitator tell their 2-minute story of self as an example.** **10 min.**
images (sights, sounds, smells, or emotions of the moment)? How did those details make you feel? - What could the storyteller do to more effectively convey why they are called to leadership in this campaign?

4. As a team **go around the group** and tell your story one by one. For each person: - 2 minutes to tell their story - 3 minutes to offer feedback from the group *Make sure your timekeeper cuts you off. This encourages focus and makes sure everyone has a chance to tell their story. Remember, the purpose here isn't to tell a perfect story, it's to practice narrative as part of the work of leadership. | 30 min |

WORKSHEET

DEVELOPING YOUR STORY OF SELF

Before you decide what part of your story to tell, think about these questions:

1. What will I be calling on others to do?
2. What values move me to take action and might also inspire others to similar action?
3. What stories can I tell from my own life about specific people or events that would *show* (rather than tell) how I learned or acted on those values?

What are the experiences in your life that have shaped the values that call you to leadership in this campaign?

FAMILY & CHILDHOOD	**LIFE CHOICES**	**ORGANIZING EXPERIENCES**
Parents/Family	School	First Experience of organizing
Growing Up	Career	Connection to key books or people
Your Community	Partner/Family	Role Models
Role Models	Hobbies/Interests/Talents	
School	Finding Passion	
	Overcoming Challenge	

Think about the challenge, choice and outcome in your story. The outcome might be what you learned, in addition to what happened.

Try drawing pictures here instead of words. Powerful stories leave your listeners with images in their minds that shape their understanding of you and your calling_._

CHALLENGE	**CHOICE**	**OUTCOME**

COACHING TIPS:

STORY OF SELF

Remember to balance both positive and constructive critical feedback. The purpose of coaching is to listen to the *way* stories are told and think of ways that the storytelling could be improved.

DON'T simply offer vague "feel good" comments. ("That was a really great story!")

DO coach each other on the following points:

1. ***THE CHALLENGE:***What were the specific challenges the storyteller faced? Did the storyteller paint a vivid picture of those challenges?
- *"When you described* _______, I got a clear picture of the challenge."_
- *"I understand the challenge to be* _______. Is that what you intended?"_
1. ***THE CHOICE:***Was there a clear choice that was made in response to each challenge? How did the choice make you feel? (Hopeful? Angry?)
- *"To me, the choice you made was* ______, and it made me feel _______."_
- *"It would be helpful if you focused on the moment you made a choice."*
1. ***THE OUTCOME:***What was the specific outcome that resulted from each choice? What does that outcome teach us?
- *"I understood the outcome was* ______, and it teaches me _______. But how does it relate to your work now?"_
1. ***THE VALUES:***Could you identify what this person's values are and where they came from? How? How did the story make you feel?
- *"Your story made me feel* _______ because _________."_
- *"It's clear from your story that you value* ______; but it could be even clearer if you told a story about where that value comes from."_
1. ***DETAILS:*** Were there sections of the story that had especially good details or images

(e.g. sights, sounds, smells, or emotions of the moment)?

- *“The image of* _______ really helped me identify with what you were feeling.”_
- *“Try telling more details about* ______ so we can imagine what you were experiencing.”_

Record Feedback/Comments from Your Team Members Here:

[Section for writing]

Coaching Your Team's “Story of Self”

As you hear each other's stories, keeping track of the details of each person’s story will help you to provide feedback and remember details about people on your team later. Use the grid below to track your team's stories in words or images.

NAME	VALUES	CHALLENGE	CHOICE	OUTCOME

ADDITIONAL RESOURCES:

Videos:

- Barack Obama, Keynote Address, “The Audacity of Hope”, Democratic National Convention, July 27, 2004, Boston, Massachusetts (first 7 minutes).
- NOI Video resource center: Story of Self

Readings:

- Jerome Bruner, “Two Modes of Thought”, Chapter 2 in Actual Minds, Possible Worlds (Cambridge: Harvard University Press, 1986), p.11 – 25.
- Martha Nussbaum, “Emotions and Judgments of Value”, Chapter 1 in Upheavals of Thought: The Intelligence of Emotions, (New York: Cambridge University Press, 2001), (pp. 19-33).
- George Marcus, The Sentimental Citizen: Emotion in Democratic Politics, (University Park: Penn State University Press, 2002), Chapter 4, “Becoming Reacquainted with Emotion” (pp.49-78)
- Malcolm Gladwell, “Small Change: why the revolution will not be tweeted”, in The New Yorker, October 4, 2010. (http://www.newyorker.com/reporting/2010/10/04/101004fa_fact_gladwell)
- Ben Brandzel, “What Malcolm Gladwell Missed About Online Organizing and Creating Big Change”, in The Nation, November 15, 2010. (http://www.thenation.com/article/156447/what-malcolm-gladwell-missed-about-online-organizing-and-creating-big-change) # TRAINERS NOTES: STORY OF SELF

Source: NOI's Story of Self Materials

Key Teaching Points: - Stories communicate values through emotions: emotions move us to act - Certain emotions inhibit action while others facilitate action: we can use stories to elicit emotions that facilitate action. - Good stories have a structure => challenge, choice, outcome - Public Narrative is a particular type of story with three parts: Self, Us, Now - Story of Self is a practice organizers can use to build community by introducing who they are and why they are called to this work. We can teach through the script of our own lives.

| SECTION | SLIDE | ANNOTATED NOTES | | ----------- | --------- | ------------------- | | Total Session Time: 80 min | | Introduction: 30 min | | TRAINER STORY (2 minutes 30 seconds) | | Tell your 2 min story of Self, as it relates to this particular training group or campaign. Use the Story of Self worksheet in the participant guide to help develop your story. | | **LEADERSHIP (30 seconds)** | | Go back to our definition of leadership. How does public narrative fit into this way of practicing leadership? Stories help us learn to make choices in the face of uncertainty. | | **WHY DO WE TELL STORIES? (1 minute)** | | ***Why do we tell stories?*** We tell stories to motivate others to take action with us. We tell stories to enable others to understand who they are and why to take action. Story telling is a leadership art that "enables others" to understand their individual, community and immediate calling. A good story helps us make choices in our own lives, by illustrating how a particular character overcame a challenge. | | **TWO WAYS OF KNOWING (1 minute)** | | There are 2 ways in which we understand the world around us: 1. The heart understands the 'why' (story); 2. The head understands the 'how' (strategy). Both lead to action, but we need to learn to lead with the head AND the Heart. Organizing without a strategy isn't driven and doesn't lead us anywhere, like walking blind. But having the best strategy in the world doesn't matter if we're not motivated to implement it and we can't get anyone to join us._ **Audience Engagement**: *By show of hands, who here has been in an organizing situation that was all about head (strategy)? Like "lets get the petitions signed and then the meeting with the congressional members and then lets make calls, etc." How'd it go? How about an organization that was all heart- storytelling, community building and no strategy? Emphasize the need for the head* ***and*** *the heart.* | | **EMOTIONS, VALUES, ACTION (1 minute)** | | Stories move us to action by engaging our emotions. Emotions tell us what we value in the world. What are the things that bring you joy & hope? What makes you feel angry? When we hear stories that make us feel a certain way those stories remind us of our core values. We experience our values through emotions. Then we are prepared to take action on those values. **Audience Engagement**: When I told my story earlier, how did it make you feel? What specific emotions did you feel? When? Why? What values did you hear in my story? Would it have been the same if I told you " I value [insert value]" How did you experience those values? | | **ACTION MOTIVATORS & INHIBITORS (2 minutes)**| | There are emotions that inhibit action and others that motivate us to act. Inertia, (no forward progress) Apathy (without caring), Fear, Isolation (fail to appreciate interest we share with others) & Self-doubt (I can't do it)– *explain the feelings. Audience Engagement (Questions to Consider): * Has anybody felt these emotions before? Have they inhibited you from taking action? Have you seen a community with some of these feelings? What are the actions of a community that has these feelings? How do these feelings change? Just by saying, "don't be afraid" or "you should care about this!" * Our emotions of inaction don't go away because some one tells us they should, but because we start to feel other emotions. * Have you been in a moment that your community felt with URGENCY? Has anyone felt that in your communities – ANGER? Have you felt HOPE, SOLIDARITY AND YOU CAN MAKE A DIFFERENCE (YCMAD)? When specifically? * What gives you that Hope, Solidarity, sense of YCMAD? | |* ***STORY STRUCTURE (1 minute)*** *| | Audience Engagement: What makes a story? Stories have a common structure. A Plot: something happens to the character, and then the character needs to make a choice, then that choice yields an outcome – and that outcome* *teaches a moral. What happens when there's no challenge? What happens when the story is all challenge and there's no outcome, or hope? | |* ***3 PARTS OF PUBLIC NARRATIVE (1 minute)*** *| | In this session you will be learning how to tell your Story of Self, but since organizing isn't just about your, but about you in relation to others, you will also be learning the two other parts of public narrative: the Story of US and Now. Three parts to public narrative:* ***1.*** *Story of Self (why I'm called to this work);* ***2.*** *Story of Us (why we're called, why we matter in this effort);* ***3.*** *Story of Now (what we're called to do right now) | |* ***SAMPLE STORY- BARACK OBAMA (15 min total to include 3 min set up, 7 min video and 5 min debrief)*** *|| Show DNC 2004 Convention Speech Trainer Tip: Because your audience may have differing political opinions, make sure to explain that we didn't choose this video because Obama is a Democrat or the President, we use this video because he is one of the greatest speakers in our country and he moved many people to action. This a craft he learned and we are here to teach you the same craft. | | | Video Debrief | Debrief: Audience Engagement: How do you feel after hearing that? What specific stories or examples painted the challenge and choice for you? What did you want to know more of? Would it have had the same impact if he had said " My name is Barack Obama, I went to Harvard Law school where I learned x and y. Then I was ran for Illinois State Senate, and won. I really support John Kerry because of x,y,z and you should too?"* ***Where's the story of self? 1*** *Challenge, choice, outcome - focus on choice points (father's journey to US, parents decision to marry, naming "Barack").* ***2*** *Notice that there is nothing about John Kerry in his speech, but rather a story that tells us about his own values. Then the audience trusts him and wants to support who he is supporting.* ***3.*** *What's the transition? ("My story is part of the larger American story")* ***Where's the story of us? 1.*** *Founding story, Declaration of Independence (why did he choose this?)* ***2.*** *Challenge, choice, outcome, hope.* ***3.*** *What's the transition? "We have more work to do"* ***Where's the story of now? 1.*** *Challenge, choice, outcome, hope.* ***2.*** *Not statistics, but stories, no 10-point plan | |* ***LEARNING TEAMS EXPLANATION (5 minutes)*** *| | Review the Story of Self worksheets and learning team agenda before the groups break out. Trainer Tip: Remind the participants to*focus on 1 or 2 choice points in your life that teach others about what values move them to be here, and like a good movie their story should be constructed so we can SEE it, HEAR it, FEEL it because of the details. | | **ROLEPLAY** | | **BREAK INTO TEAMS 65 min** | All materials are in the participant guide | **1.** Review agenda, Set Norms, assign a time keeper (10 min) **2.** Develop your story (10 min) **3.** Partner pair-and-share (15 min) **4.** Share stories in small group (30 min) | | **DEBRIEF 20 min** | | Invite 2-3 participants to tell their stories. Have group coach them on making their stories better. Also, define what makes for effective coaching. What coaching questions helped your stories improve? **1.** Did they tell a real Story of Self with a clear moment(s) of choice **2.** Did they avoid the trap of just talking about the campaign? If not how could they? What in their life are you still curious about? **3.** Was there good detail? Where could there be more? **4.** Coaching—what types of feedback & questions were helpful? | # Top 5 Open Source Project Management Tools (in 2014)

Source OpenSource.com. Also, reference 2014's posting.

Last year, I covered five of the best open source project management tools, like ProjectLibre and OpenProject. The article struck a chord with readers and continues to prove valuable. So, this year I revisited the tools mentioned in last year's article, taking into account comments and suggestions from readers, and provided an update on where they are today. First, I share five new open source project management tools for 2015. All in all, this article will give you a good look at 11 of the top open source project management tools out there.

The criteria I used is based on the following: - Is the software provided under an open source license? - Does it have an active community? - Does it have up-to-date documentation available? - Is the source code available? - Are there new or recent releases?

Five new tools for 2015

Tuleap Open ALM

Tuleap Open ALM is not just a project management tool, it is an application lifecycle management tool, including support for agile development and project management. Tuleap received the InforWorld.com Bossie Award in 2013 and is used by Fortune 500 companies, small and medium businesses, and open source projects.

Tuleap allows for agile, traditional, hybrid, or custom processes for project magament. It supports planning, sprints, tasks, reports, and more. This tool is very suitable for open source development companies, as the tool also integrates with Git, SVN, Jenkins, and more.

Tuleap is licensed under a GNU Public License and is available on GitHub. The open source version contains all features and allows for unlimited projects and users. In addition to professional support, there is an active community for support and documentation. Tuleap is in active development, with version 7.8 in December 2014 being the latest release.

OrangeScrum

OrangeScrum is a modern project management tool for freelancers, agencies, and small and medium businesses. Features include a scrum task board, resource planning, progress tracking, and more. It is designed for use by IT companies, education and health services, construction and manufacturing, and others.

The tool is web-based on top of a CakePHP framework. It is available as a free and open source under GPLv3 and available on GitHub. There is also a paid SaaS solution in which the parent company Andolasoft provides professional services and support.

The OrangeScrum developer community has an active forum and roadmap. The latest version is 1.5.1 and under active development.

Taiga

Taiga is an open source project management platform in beta for startups, agile developers, and designers. For an introduction, see Nitish Tiwari's coverage in *Taiga, a new open source project management tool with focus on usability*. Read more in this interview with CEO Pablo Ruiz Múzquiz.

"Taiga is a tool that aims to solve the basic problem of software usability. Designed with this sole aim, the developers claim it's beautiful to look at all day long."

With a focus on agile development, Taiga has all of the required features such as a backlog, Kanban, tasks, sprints, and issues. Taiga is open source under the GNU Affero GPLv3 and available on GitHub. There is also a paid version. Taiga runs on Nginx, Python 3.4, and PostgreSQL 9.3 or higher. You can find documentation on their site and support via their Google group.

Odoo

Odoo, formerly known as OpenERP, is a full suite of business applications, of which Odoo project management is just one. It is multiplatform and supports Windows, Fedora, CentOS, RHEL, and others. Odoo is licensed under AGPLv3 and available for download.

Odoo has a professional community with everything a user or developer needs, including forums, documentation, IRC, tracker, GitHub, and more. It supports a Kanban view with tasks and issues as well as a Gantt chart and control deadlines. The tool can easily be adapted to meet your own project management methods. It's also highly collaborative, allowing for multiple members to work on proposals or minutes at the same time and sending tasks as emails automatically. For the full set of features, check out the tools description page.

MyCollab

MyCollab supports three modules: MyCollab CRM, Document Management, and MyCollab-Project.

MyCollab-Project is the tool we're looking at in this case and has many features, like a Gantt chart and milestones, to time tracking, issue management, risk and problem management. MyCollab-Project comes in three editions, of which the Community Edition is the free option. MyCollab is licensed under AGPLv3, built in Java, and bundles open source frameworks and libraries. The source code is on GitHub and the tool is available for download on the website. Documentation is available there as well.

MyCollab-Project's latest release is 4.5.5 with a focus on enhancing project functionalities and mobile distribution. GitHub is used in place of a traditional forum, where they accept contributions and provide support.

Progress report on tools featured in 2014

Here's an update on the projects I covered last year.

ProjectLibre has moved from version 1.5.8 to 1.5.9. It has not yet released version 2.0, nor is there a SaaS (software as a service) option available. In July 2014, ProjectLibre passed the 1 million mark for number of downloads.

LibrePlan current version is the one they released in April 2013. For a full update, check the State of the Union 2015.

OpenProject is active and continues to improve as evidenced by their release notes. They are currently at version 4.0.4. The migration to Ruby on Rails 2.1 and 3.2 was successfully completed in March 2014 with their release of version 3.0.

project-open released a 4.1 beta in June 2014. The expected 4.2 has not been released yet, but I spotted a version 5.0 on their roadmap for 2015.

Redmine seems to continue with steady improvements.

The bonus tool I covered last year was Agilefant. They have since implemented a paid-for version but continue to provide the basic tool as open source.