

# USING GIT ON OS-X

Munich Cocoaheads 2009-11-12

©2009 Stephen Riehm

# COMING UP

- Basic Concepts
- Daily Git
- Git with XCode
- ~~• Non-Obvious Git~~

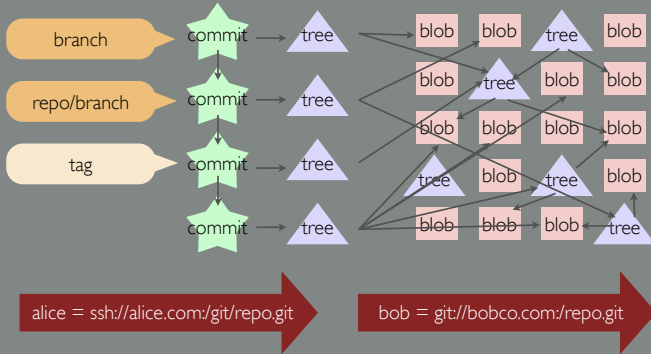
# WHAT IS GIT?

project\_dir/

.git/

# WHAT IS GIT?

.git/



# BLOBS

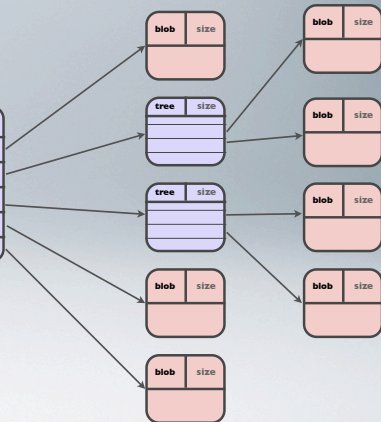
58206c1725109d441fe846300fd4e57063cc6d6b

| blob   | size |
|--|------|
| <pre>// Blah.m // This class does  @implementation Bla  @synthesize a;</pre> |      |

# TREES

56906c1...

| tree      |      | size     |            |
|-----------|------|----------|------------|
| rw-rw-rw- | blob | 5b1d3... | ReadMe     |
| rw-rw-rw- | tree | 03e78... | Classes    |
| rw-rw-rw- | tree | cdc8b... | Resources  |
| rw-rw-rw- | blob | cba0a... | Info.plist |
| rw-rw-rw- | blob | 911e7... | distrib.pl |



# CHANGES

3500621...

| tree      |      | size     |            |
|-----------|------|----------|------------|
| rw-rw-rw- | blob | 5b1d3... | ReadMe     |
| rw-rw-rw- | tree | 03e78... | Classes    |
| rw-rw-rw- | tree | cdc8b... | Resources  |
| rw-rw-rw- | blob | cba0a... | Info.plist |
| rw-rw-rw- | blob | 911e7... | distrib.pl |



|      |      |
|------|------|
| blob | size |
|------|------|

|      |      |
|------|------|
| blob | size |
|------|------|

|      |      |
|------|------|
| tree | size |
|------|------|

|      |      |
|------|------|
| blob | size |
|------|------|

|      |      |
|------|------|
| tree | size |
|------|------|

|      |      |
|------|------|
| blob | size |
|------|------|

|      |      |
|------|------|
| blob | size |
|------|------|

|      |      |
|------|------|
| blob | size |
|------|------|

|      |      |
|------|------|
| blob | size |
|------|------|



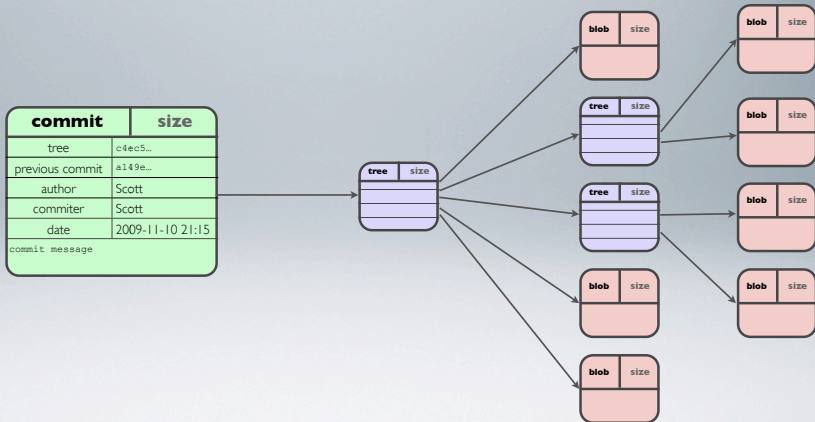
# COMMITTS

56906c1...

| commit          | size             |
|-----------------|------------------|
| tree            | c4ec5...         |
| previous commit | a149e...         |
| author          | Scott            |
| committer       | Scott            |
| date            | 2009-11-10 21:15 |
| commit message  |                  |



# COMMITS

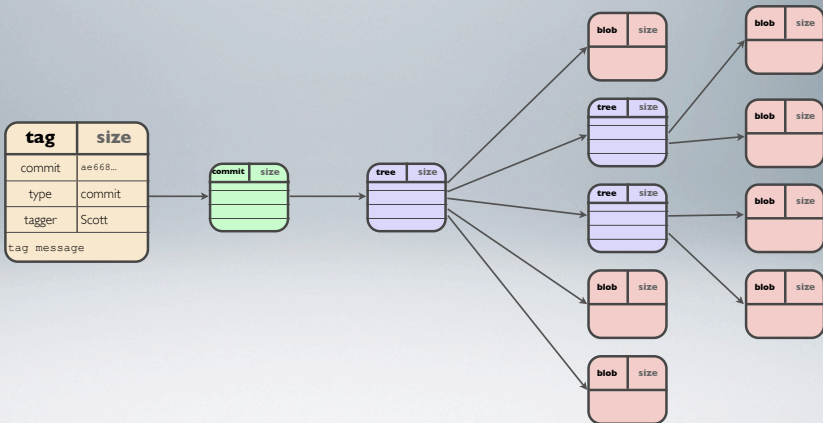


# TAGS

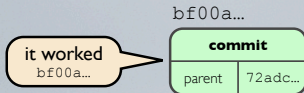
56906c1...

| tag         | size     |
|-------------|----------|
| commit      | ae668... |
| type        | commit   |
| tagger      | Scott    |
| tag message |          |

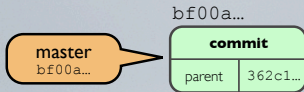
# TAGS



# LIGHT-WEIGHT TAGS



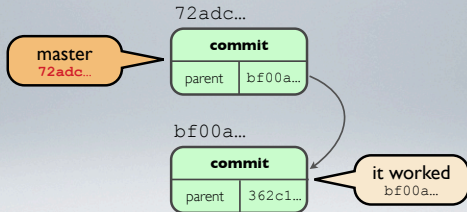
# BRANCHES



# BRANCHES 'V' TAGS



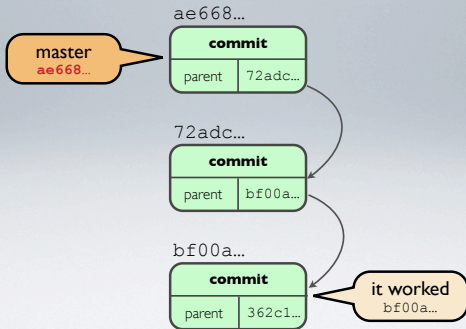
# BRANCHES 'V' TAGS





# BRANCHES 'V' TAGS

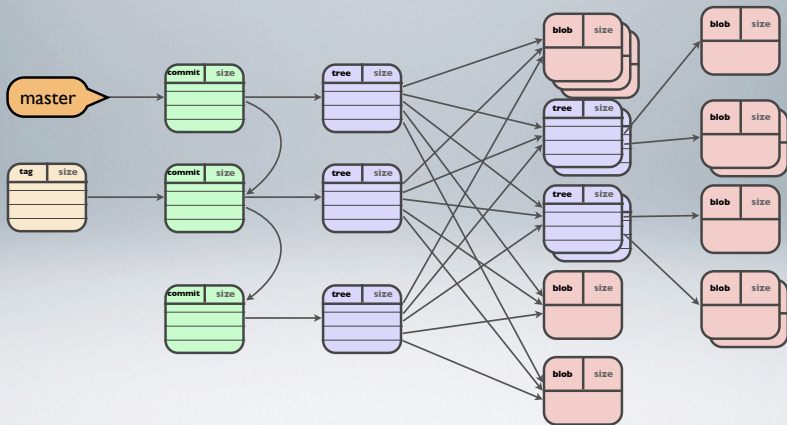
HEAD and your current branch are automatically updated when you commit a change.



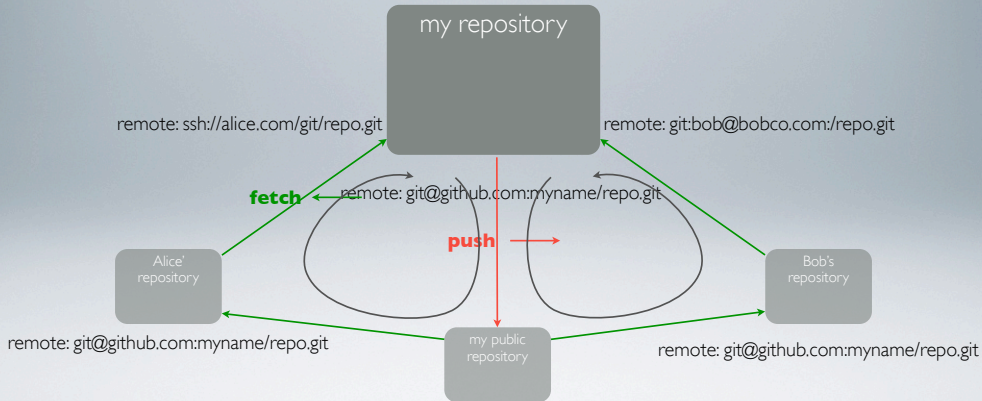
tags are immutable and reliably point to a known state of the entire repository



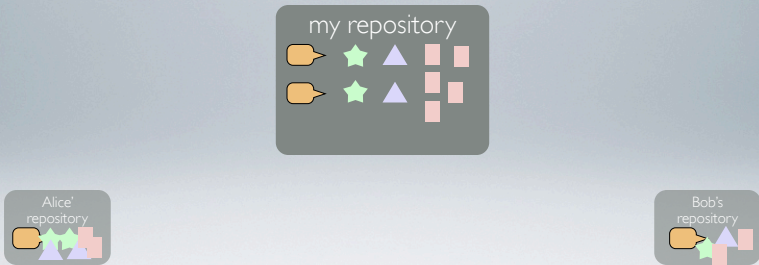
# THE WHOLE LOT



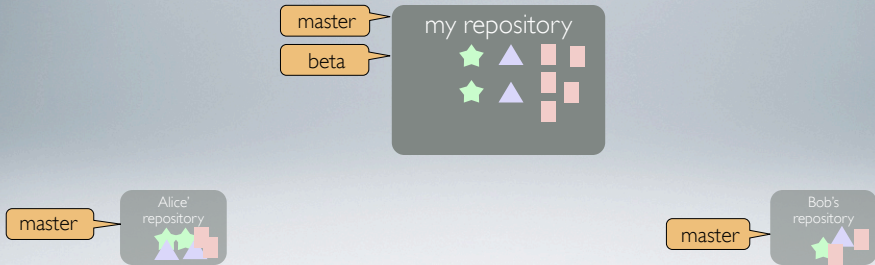
# SHARING



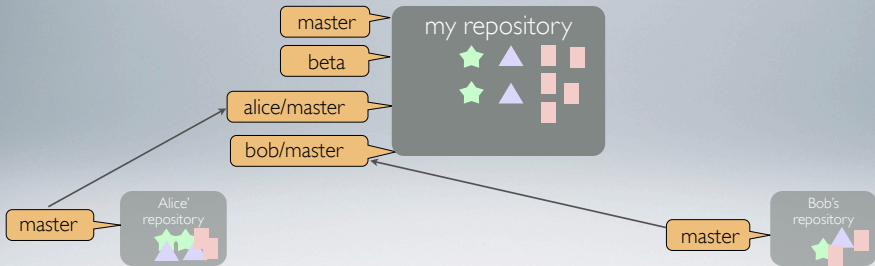
# NAMESPACES



# NAMESPACES

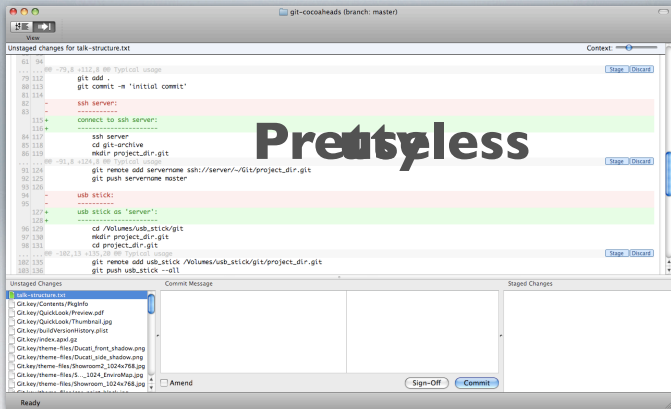


# NAMESPACES

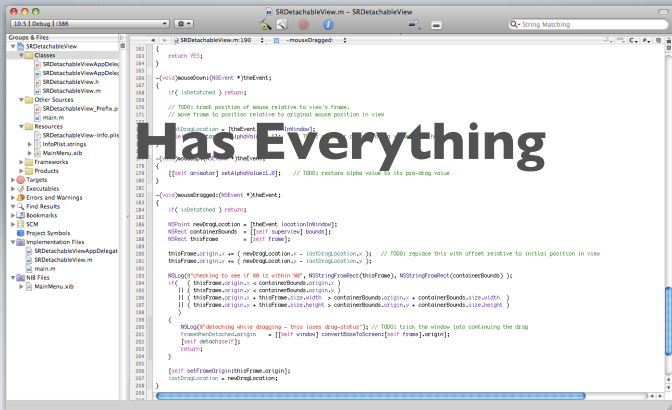


# GIT ON OS-X

# GITX

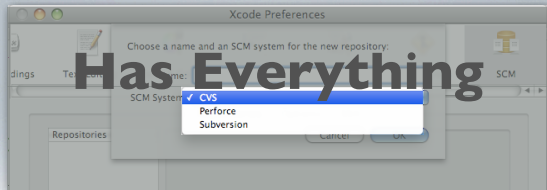


# XCODE





# XCODE



# GIT YOUR HANDS DIRTY

```
~/ > echo "you're going to need the command line :-)"
```

# FIRST STEPS

# GLOBAL CONFIGURATION

```
~/ > $EDITOR ~/.gitconfig
```

```
[core]
  pager = more
  excludesfile = /Users/me/.gitignore
[user]
  name = My Full Public Identity
  email = h4x0r@example.com
[format]
  pretty = format:%h %ci [%aN] %s
```

pager stops long lists from flying past your nose uncontrollably

excludesfile specifies files should never be checked into a git repository

format:

%h = hash id

%ci = commit date, iso 8601 format

%aN = author name

%s = summary

more info: `git log --help`

# GLOBAL CONFIGURATION

```
~/ > $EDITOR ~/.gitignore
```

```
# apple typical files
.DS_Store
.Spotlight-V100
.com.apple.timemachine.supported
.fseventsdbuild

# XCode user state files
*.modelv3
*.pbxuser
*.objc_sync

# other SCM systems
.svn

# editor temporary files
*~
*.swp

# files you generate while building
build/
version.txt
CHANGELOG
```

# WHERE TO LOOK FOR HELP

```
~/ > git help <cmd>
```

```
~/ > git <cmd> --help
```

make sure you install  
git's man-pages.

Use the [man](#) branch of  
the git repository

# WHERE TO LOOK FOR HELP

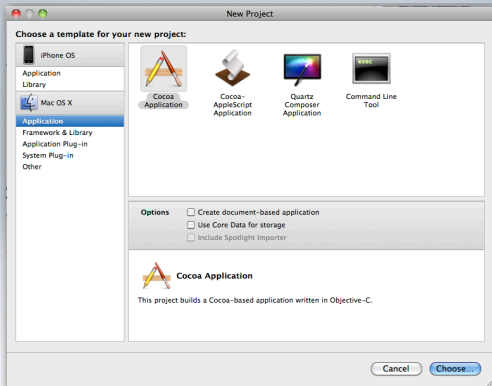
<http://git-scm.com>

<http://github.com>

<http://gitready.com>

<http://google.com>

# A NEW PROJECT





# NEW PROJECT IN GIT

```
~/> cd project
```

```
~/project/ > git init
```

```
Initialized empty Git repository in project_dir/.git/
```

```
~/project/ > git add .
```

```
~/project/ > git commit -m 'initial commit'
```

```
[master (root-commit) 64fb323] initial commit
```

```
1 files changed, 1 insertions(+), 0 deletions(-)
```

```
create mode 100644 hello.txt
```

create your project in XCode  
create a new git repository  
add your files & directories to git  
commit your changes

# GIT CONFIGURATION FOR XCODE

```
~/project/ > $EDITOR .gitattributes
```

```
*.pbxproj -crlf -diff -merge
*.nib      -crlf -diff -merge
*.xib      -crlf -diff -merge
*.graaffle -crlf -diff -merge
```

```
~/project/ > git add .gitattributes
```

```
~/project/ > git commit -m 'add .gitattributes - prevent accidental merging of special XCode files'
```

```
[master (root-commit) 64fb323] initial commit
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 .gitattributes
```

Tell git to treat these files as if they were binaries.

XML files are notorious for being text, but "unmergeable".

`.gitattributes` is project-specific  
must be checked into each project separately  
will automatically be used by all project members

# JOINING AN EXISTING PROJECT

cloning a repository automatically sets up a remote repository called `origin`.

You can specify a different name for the remote repository with `-o name`

```
~/ > git clone -o cloned_repo URL/project.git
```

```
~/ > cd project
```

```
~/project/ > git checkout -b my_stuff cloned_repo/master
```

# CLONE A LOCAL REPOSITORY

cloning a repository automatically sets up a remote repository called `origin`.

You can specify a different name for the remote repository with `-o name`

```
~/ > git clone ~/old_project_dir ~/new_project_dir
```

DAILY WORK

# ADD & COMMIT

work work work...

```
~/project/ > git status
```

```
~/project/ > git add file file file directory... or git add -A or git add -u
```

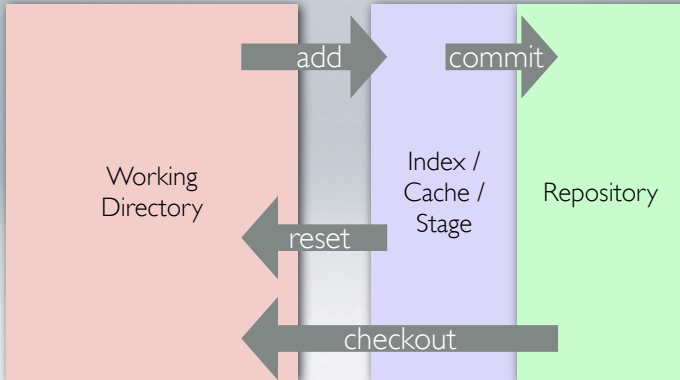
```
~/project/ > git status
```

```
~/project/ > git commit -m 'what I just did'
```

`git add -A`  
new files  
changed files  
removed files

`git add -u`  
changed files  
removed files

# WHY ADD & COMMIT?



# COMMIT (WITHOUT ADD)

git commit -a  
same as  
git add -A; git commit

```
~/project/ > git commit -a -m 'what I just did'
```



# BRANCHING

```
~/project/ > git checkout -b new_branch
```

```
~/project/ > git branch -a
```

```
~/project/ > git branch -d old_branch
```

```
~/project/ > git branch -D old_branch
```

TIP: you can create a new branch AFTER you have already made changes.

Just `checkout -b new_branch` before you `git add`

# DIFFERENCES?

```
~/project/ > git diff
```

```
~/project/ > git diff --cached
```

```
~/project/ > git diff HEAD
```

```
~/project/ > git diff branch
```

# MERGING

git always merges into the working directory

merged files are added automatically

conflicts are not added - you need to resolve them first

```
~/project/ > git merge other_branch
```

fix conflicts...

```
~/project/ > git add -A
```

```
~/project/ > git commit -m 'merge changes from other_branch'
```

# THROWING THINGS AWAY

`git reset --hard` updates the cache and the working tree to match the named branch (by default `HEAD`)

```
~/project/ > git reset
```

```
~/project/ > git reset --hard HEAD
```

# MULTIPLE REPOSITORIES

# PUBLISHING YOUR REPOSITORY

```
local ~/project/ > ssh me@remote.com
```

```
remote ~/ > mkdir project.git
```

```
remote ~/project.git/ > cd project.git
```

```
remote ~/project.git/ > git init --bare
```

```
remote ~/project.git/ > logout
```

```
local ~/project/ > git remote add public_repo ssh://me@remote.com/~/project.git
```

```
local ~/project/ > git push public_repo release_branch
```

you should NOT publish your private directories (basic security)

create a **bare repository** on a public server

push only the branches your wish to publish

# USB-STICK

```
~/project/ > git clone --bare . /Volumes/usb_stick/project.git
```

```
~/project/ > git remote add usb_stick /Volumes/usb_stick/project.git
```

```
~/project/ > git push usb_stick
```

USB stick, external disk

great for ad-hoc sharing

great for backup

treat like a public  
repository

# REALLY QUICK SHARING

copying a repository works  
just fine - both copies can  
synchronise with each  
other!

```
~/project/ > cp -R . /Volumes/usb_stick/project
```

```
~/project/ > git remote add usb_stick /Volumes/usb_stick/project
```

```
~/project/ > git push usb_stick
```



# WHICH REPOS AM I CONNECTED TO?

```
~/project_dir/ > git remote -v  
public_repo      ssh://me@remote.com/~project.git (fetch)  
public_repo      ssh://me@remote.com/~project.git (push)  
usb_stick        /Volumes/usb_stick/project.git (fetch)  
usb_stick        /Volumes/usb_stick/project.git (push)
```

# UPDATES FROM MULTIPLE REPOS

```
~/project/ > $EDITOR .git/config
```

```
...  
[remote "steve"]  
    url = ssh://steveserve.com/~Git/project.git  
    fetch = +refs/heads/*:refs/remotes/steve/*  
[remote "mac"]  
    url = git@github.com:mac/project.git  
    fetch = +refs/heads/*:refs/remotes/mac/*  
[remotes]  
    buddies = steve mac  
...
```

```
~/project/ > git remote update buddies
```

```
Updating steve
```

```
...  
Updating mac
```

```
...
```

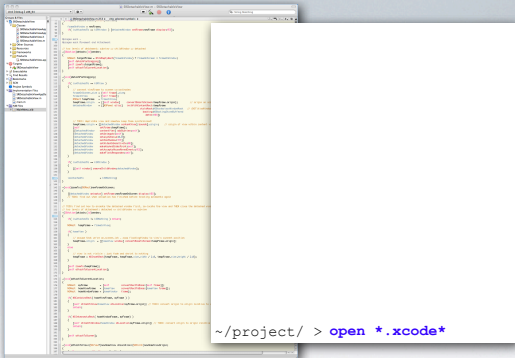
# WORKING WITH OTHERS

Publish your changes via a bare repository

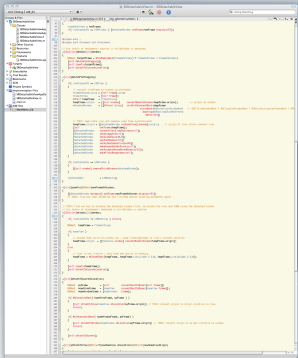
Never push to someone else's repository

Use `git remote update` to track multiple repositories

# XCODE & GIT



# XCODE & GIT



```
~/project/ > git status
```

```
~/project/ > git diff
```

```
~/project/ > git checkout -b fix
```

```
~/project/ > git commit -am '...'
```

```
~/project/ > git checkout master
```

```
~/project/ > git merge fix
```

```
~/project/ > git push public
```

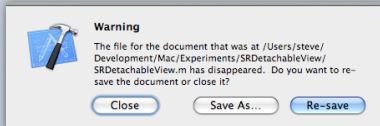
# DON'T PANIC

`git checkout`



**"Read from Disk"**, will ~~only~~ just ~~code~~ update with your repository

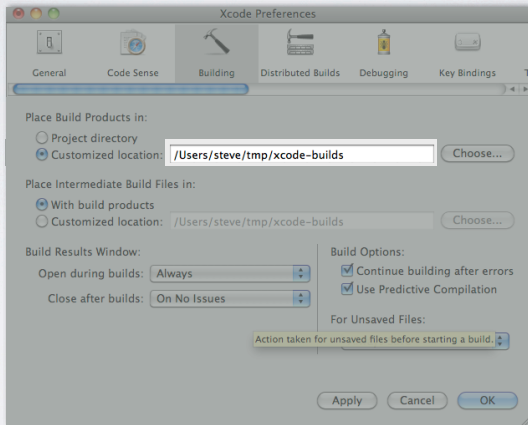
# HOWEVER...



If you get this message, you should:

- Save your work (possibly in a Temporary Directory)
- Close XCode
- Fix up your working directory
- Open XCode again

# XCODE - TIPS





# THANKS

- Patrick Stein
- Matthias Carell