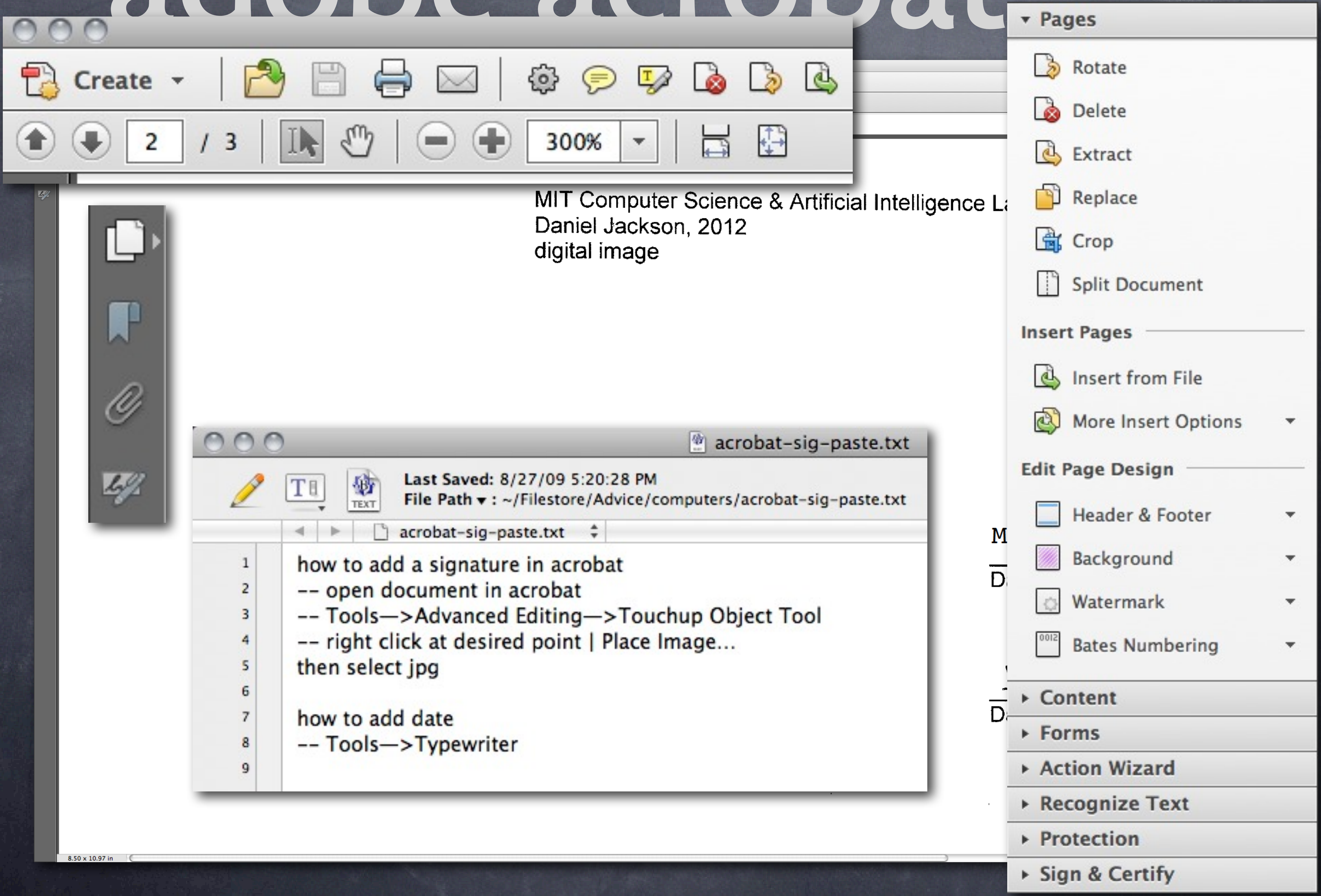
what's wrong with git?

Daniel Jackson
IFIP 2.3 · Seattle, July 2012
with Alcino Cunha, Jonathan Edwards, Eunsuk Kang and Andrea Mocci

motivations

adobe acrobat



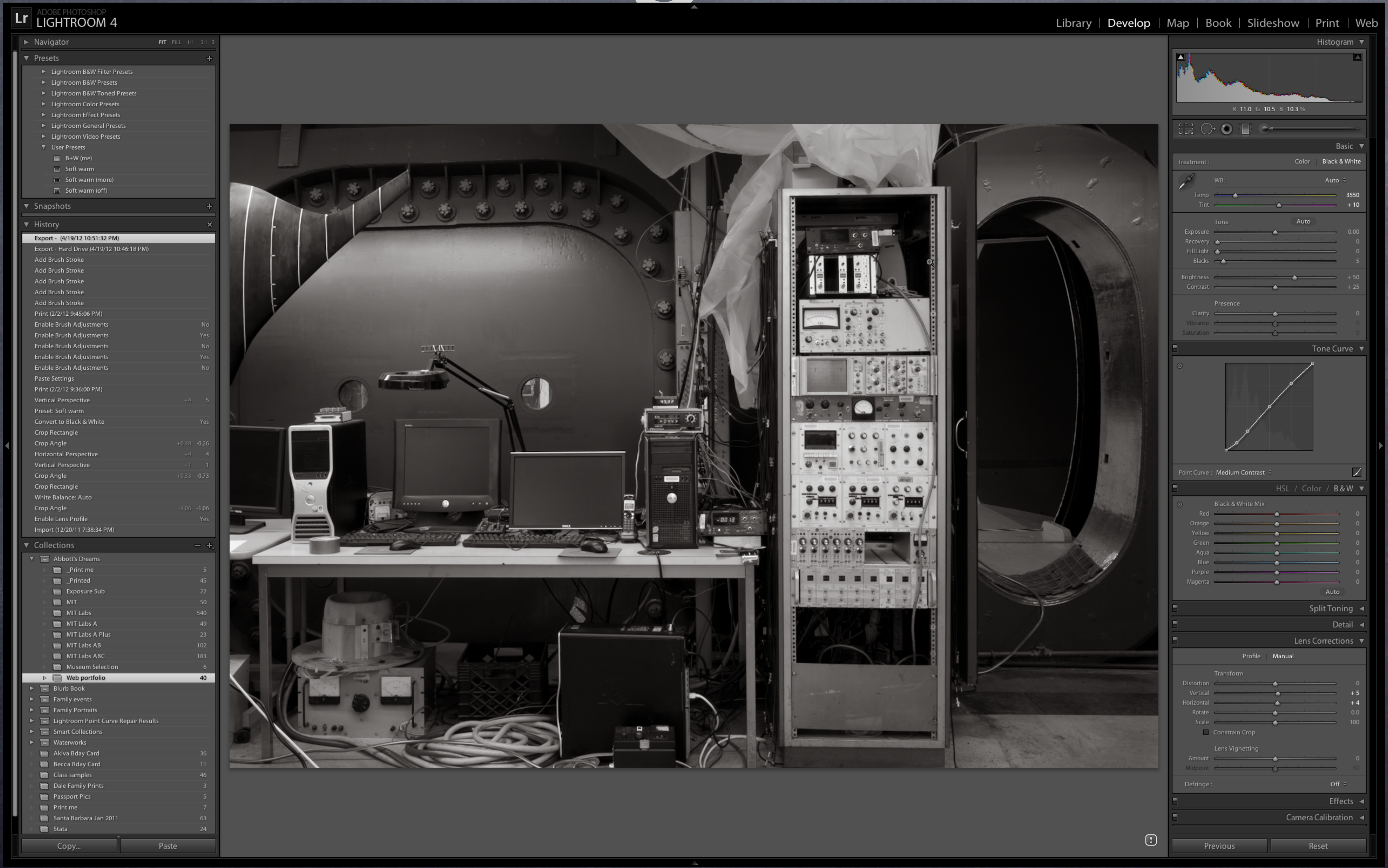
MIT Computer Science & Artificial Intelligence La
Daniel Jackson, 2012
digital image

acrobat-sig-paste.txt

Last Saved: 8/27/09 5:20:28 PM
File Path ▾ : ~/Filestore/Advice/computers/acrobat-sig-paste.txt

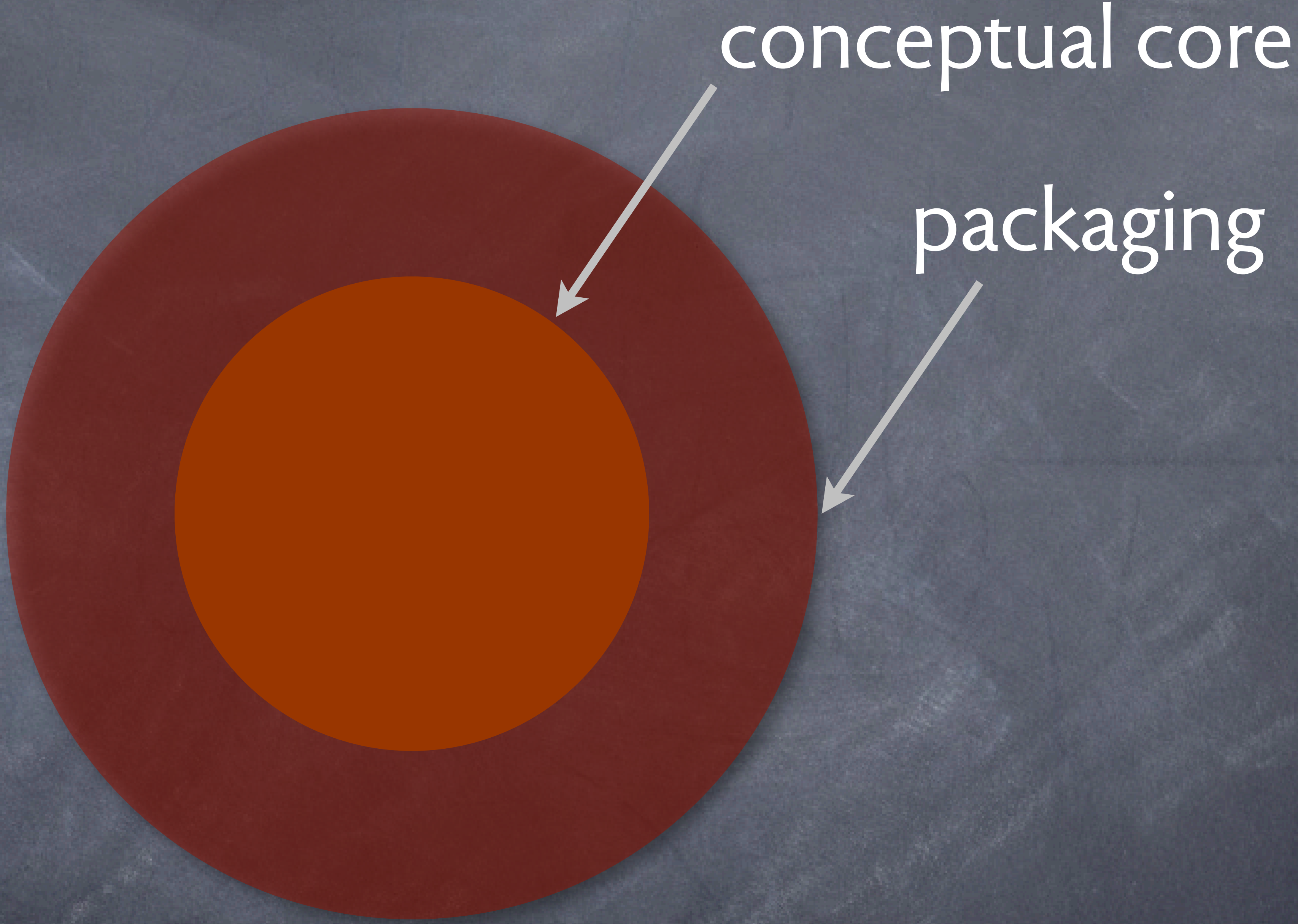
```
how to add a signature in acrobat
-- open document in acrobat
-- Tools—>Advanced Editing—>Touchup Object Tool
-- right click at desired point | Place Image...
then select jpg

how to add date
-- Tools—>Typewriter
```

adobe lightroom



```
Presets
  Lightroom B&W Filter Presets
  Lightroom B&W Presets
  Lightroom B&W Toned Presets
  Lightroom Color Presets
  Lightroom Effect Presets
  Lightroom General Presets
  Lightroom Video Presets
  User Presets
    B+W (me)
    Soft warm
    Soft warm (more)
    Soft warm (off)

History
  Export -  (4/19/12 10:51:32 PM)
  Export - Hard Drive (4/19/12 10:46:18 PM)
  Add Brush Stroke
  Add Brush Stroke
  Add Brush Stroke
  Add Brush Stroke
  Add Brush Stroke
  Print (2/2/12 9:45:06 PM)
  Enable Brush Adjustments                   No
  Enable Brush Adjustments                   Yes
  Enable Brush Adjustments                   No
  Enable Brush Adjustments                   Yes
  Enable Brush Adjustments                   No
  Paste Settings
  Print (2/2/12 9:36:00 PM)
  Vertical Perspective               +4      5
  Preset: Soft warm
  Convert to Black & White                   Yes
  Crop Rectangle
  Crop Angle                      +0.48  -0.26
  Horizontal Perspective             +4      4
  Vertical Perspective               +1      1
  Crop Angle                      +0.33  -0.73
  Crop Rectangle
  White Balance: Auto
  Crop Angle                      -1.06  -1.06
  Enable Lens Profile                        Yes
  Import (12/20/11 7:38:34 PM)

Collections
  Abbott's Dreams
    _Print me                                  5
    _Printed                                  45
    Exposure Sub                              22
    MIT                                       50
    MIT Labs                                 540
    MIT Labs A                                49
    MIT Labs A Plus                           23
    MIT Labs AB                              102
    MIT Labs ABC                             183
    Museum Selection                           6
    Web portfolio                             40
  Blurb Book
  Family events
  Family Portraits
  Lightroom Point Curve Repair Results
  Smart Collections
  Waterworks
  Akiva Bday Card                             36
  Becca Bday Card                             11
  Class samples                               46
  Dale Family Prints                           3
  Passport Pics                                5
  Print me                                     7
  Santa Barbara Jan 2011                      63
  Stata                                       24
```

```
Histogram
  R 11.0  G 10.5  B 10.3 %

Basic
  Treatment :            Color | Black & White
  WB :                   Auto
    Temp                 3550
    Tint                 + 10
  Tone                   Auto
    Exposure             0.00
    Recovery             0
    Fill Light           0
    Blacks               5
    Brightness           + 50
    Contrast             + 25
  Presence
    Clarity              0
    Vibrance             0
    Saturation           0

Tone Curve
  Point Curve :  Medium Contrast

HSL / Color / B&W
  Black & White Mix
    Red                  0
    Orange               0
    Yellow               0
    Green                0
    Aqua                 0
    Blue                 0
    Purple               0
    Magenta              0
                         Auto

Lens Corrections
  Profile | Manual
  Transform
    Distortion           0
    Vertical             + 5
    Horizontal           + 4
    Rotate               0.0
    Scale                100
    Constrain Crop
  Lens Vignetting
    Amount               0
    Midpoint             50
  Defringe :             Off
```


conceptual core
packaging

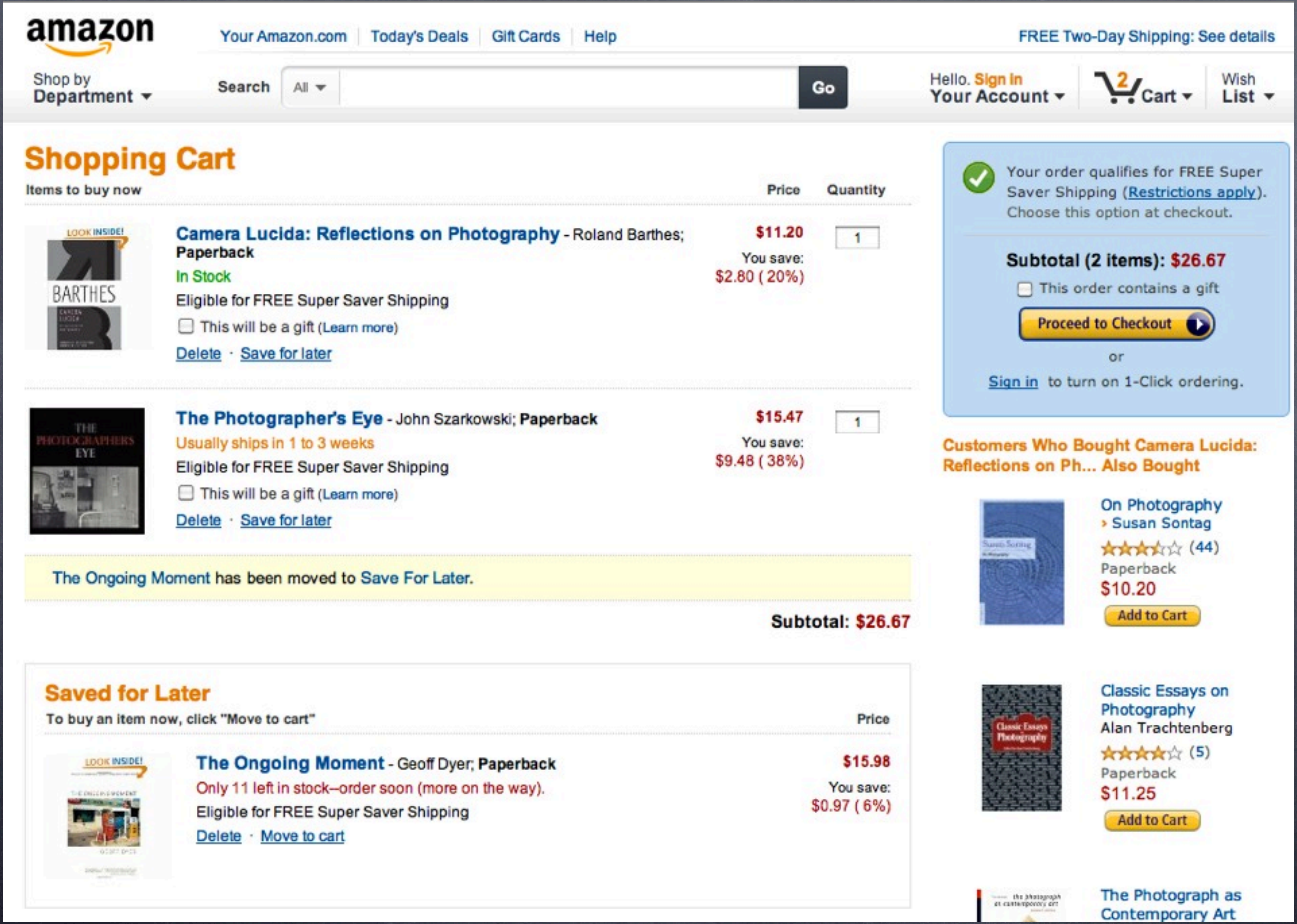
premise

agenda

The quality of an application's conceptual core determines whether it is usable, dependable and maintainable.

Develop a constructive theory of design based on conceptual models.

example: shopping cart



amazon
Your Amazon.com | Today's Deals | Gift Cards | Help
FREE Two-Day Shipping: See details
Shop by Department
Search
All
Go
Hello. Sign in Your Account
2 Cart
Wish List

Shopping Cart

Items to buy now

	Price	Quantity
Camera Lucida: Reflections on Photography - Roland Barthes; **Paperback** In Stock Eligible for FREE Super Saver Shipping ☐ This will be a gift (Learn more) Delete · Save for later	$11.20 You save: $2.80 (20%)	1
The Photographer's Eye - John Szarkowski; **Paperback** Usually ships in 1 to 3 weeks Eligible for FREE Super Saver Shipping ☐ This will be a gift (Learn more) Delete · Save for later	$15.47 You save: $9.48 (38%)	1

The Ongoing Moment has been moved to Save For Later.

Subtotal: $26.67

Saved for Later

To buy an item now, click "Move to cart"

	Price
The Ongoing Moment - Geoff Dyer; **Paperback** Only 11 left in stock--order soon (more on the way). Eligible for FREE Super Saver Shipping Delete · Move to cart	$15.98 You save: $0.97 (6%)

Your order qualifies for FREE Super Saver Shipping (Restrictions apply). Choose this option at checkout.

Subtotal (2 items): $26.67

☐ This order contains a gift

Proceed to Checkout

or

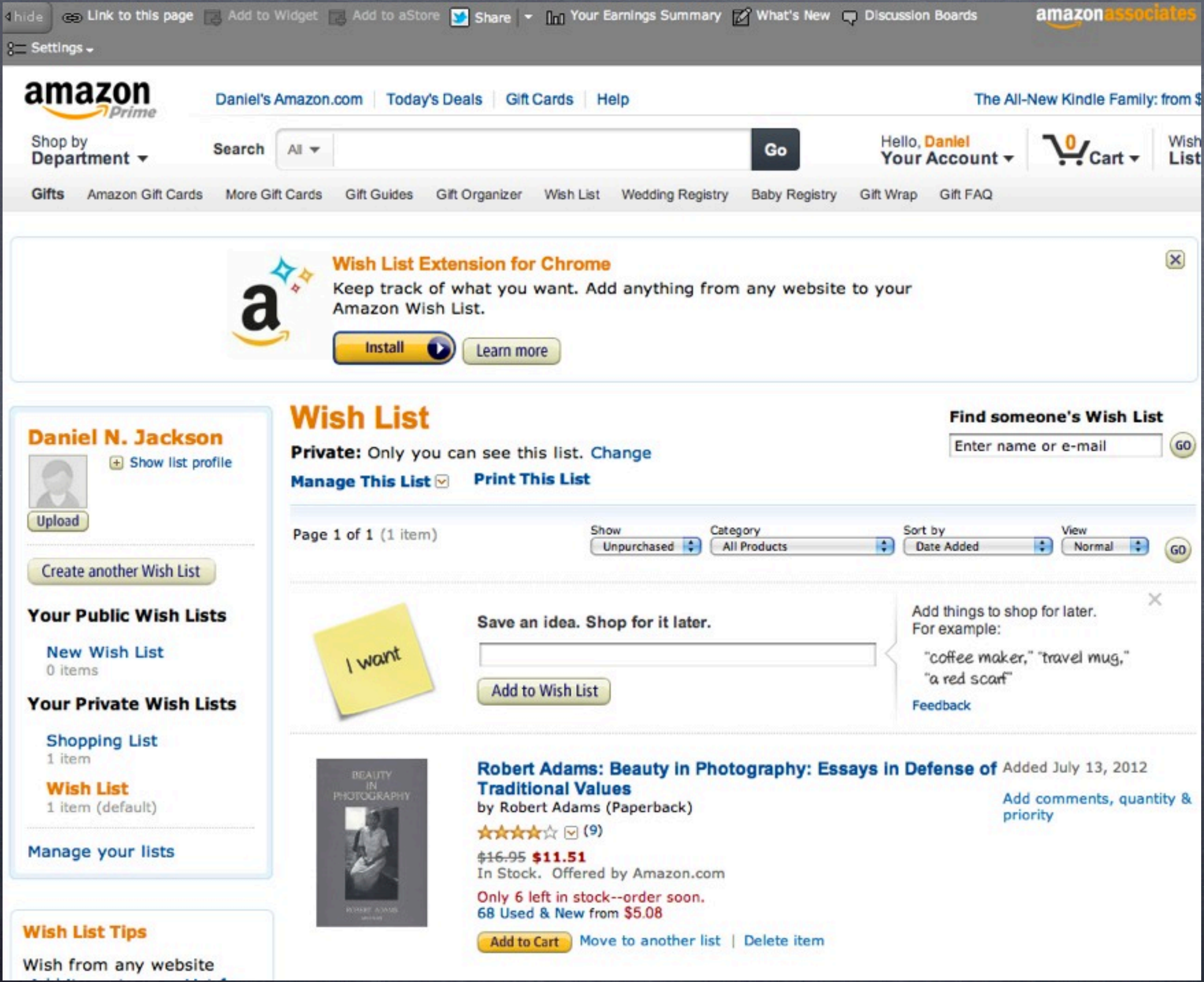
Sign in to turn on 1-Click ordering.

Customers Who Bought Camera Lucida: Reflections on Ph... Also Bought

On Photography
› Susan Sontag
(44)
Paperback
$10.20
Add to Cart

Classic Essays on Photography
Alan Trachtenberg
(5)
Paperback
$11.25
Add to Cart

The Photograph as Contemporary Art



Wish List Extension for Chrome

Keep track of what you want. Add anything from any website to your Amazon Wish List.

Install | Learn more

Daniel N. Jackson

Show list profile

Upload

Create another Wish List

Your Public Wish Lists

New Wish List
0 items

Your Private Wish Lists

Shopping List
1 item

Wish List
1 item (default)

Manage your lists

Wish List Tips

Wish from any website

Wish List

Private: Only you can see this list. Change

Manage This List | Print This List

Find someone's Wish List

Enter name or e-mail | GO

Page 1 of 1 (1 item)

Show: Unpurchased | Category: All Products | Sort by: Date Added | View: Normal | GO

Save an idea. Shop for it later.

Add to Wish List

Add things to shop for later. For example:

"coffee maker," "travel mug," "a red scarf"

Feedback

Robert Adams: Beauty in Photography: Essays in Defense of Traditional Values
by Robert Adams (Paperback)

(9)

~~$16.95~~ $11.51

In Stock. Offered by Amazon.com

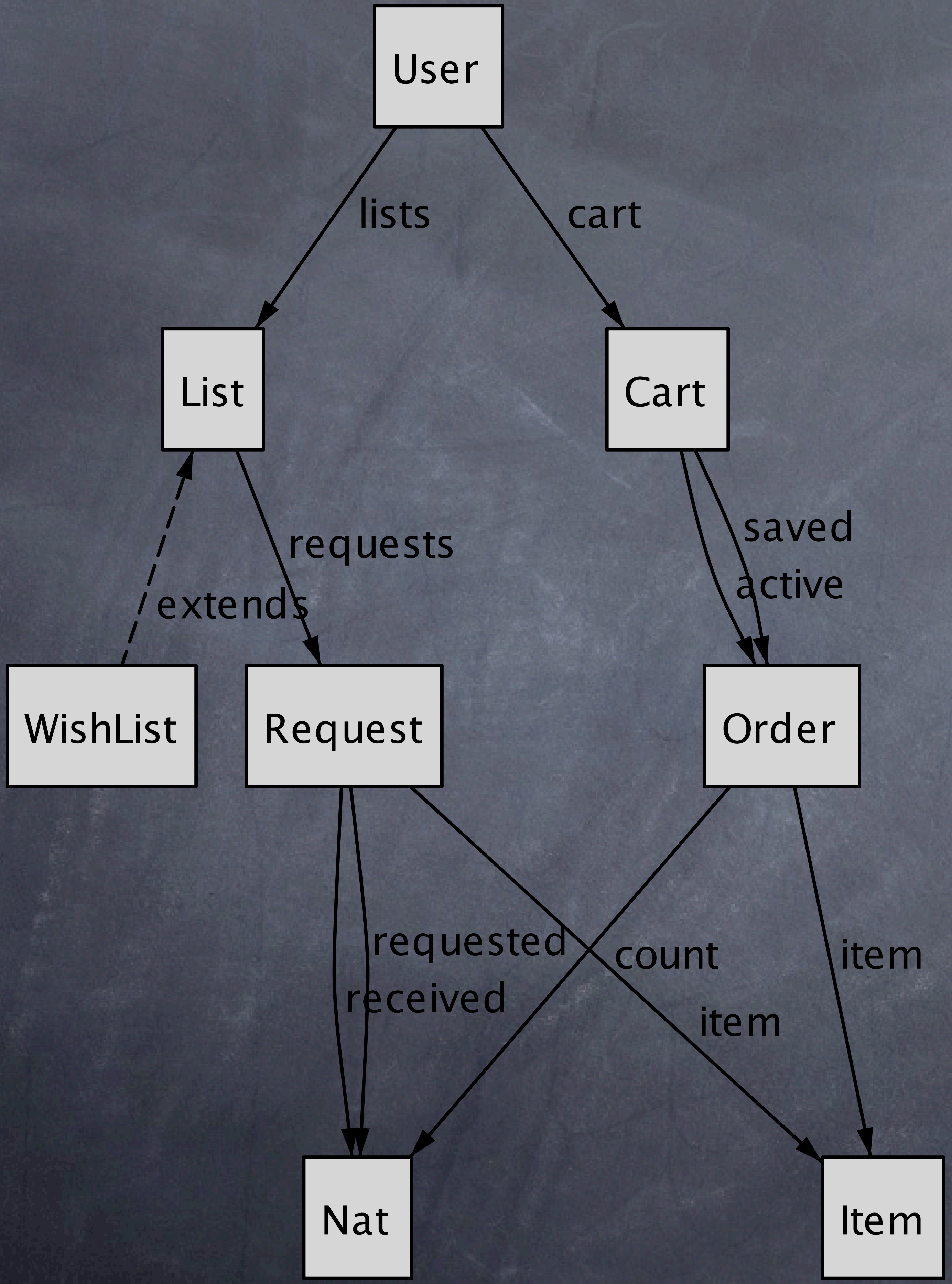
Only 6 left in stock--order soon.

68 Used & New from $5.08

Add to Cart | Move to another list | Delete item

Added July 13, 2012

Add comments, quantity & priority



operations

add item
move between cart/lists
change counts
checkout cart
create/delete lists

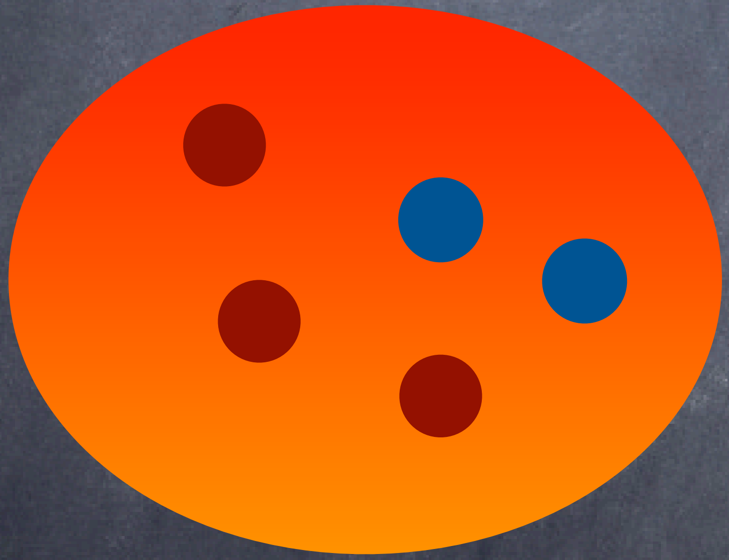
three criteria

consistent

same structure ⇒ same behavior

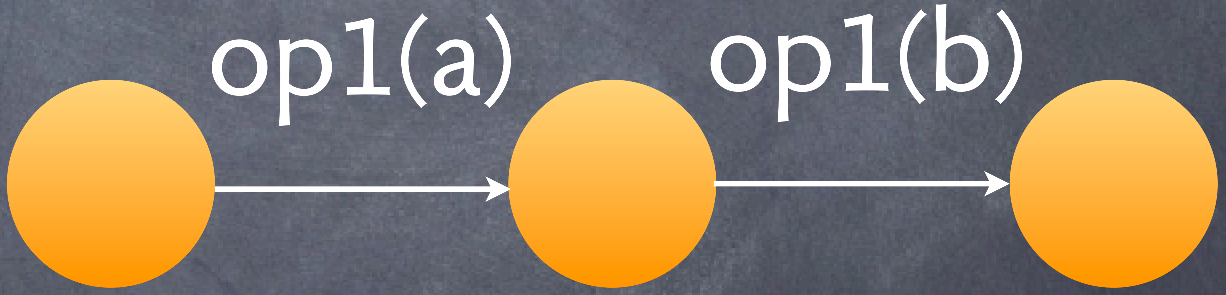
uniform

- same set, same features



coherent

- same op, same effect



functional

states & operations are accessible

visible

- can see state & available ops

realizable

- can reach desired state
- can undo operations

modular

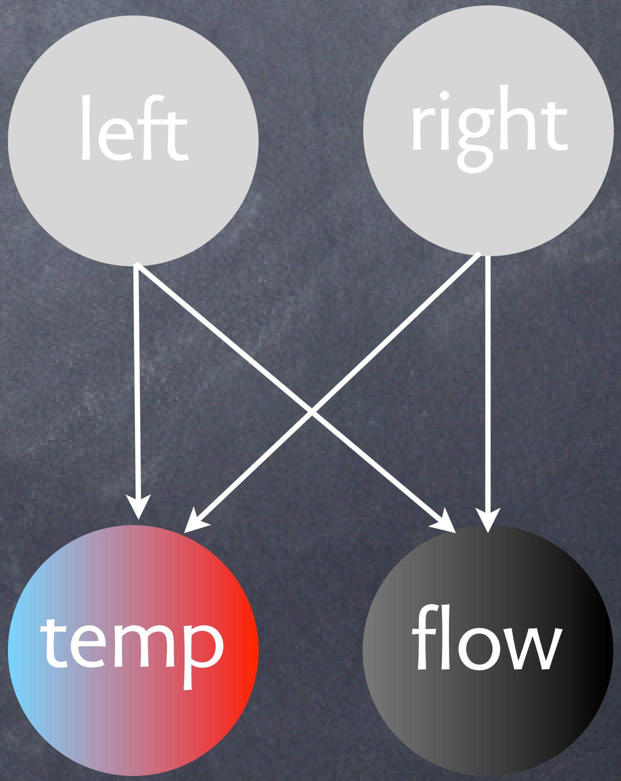
can see it and do it

decoupled

- operations don't mix features

coupled

uncoupled



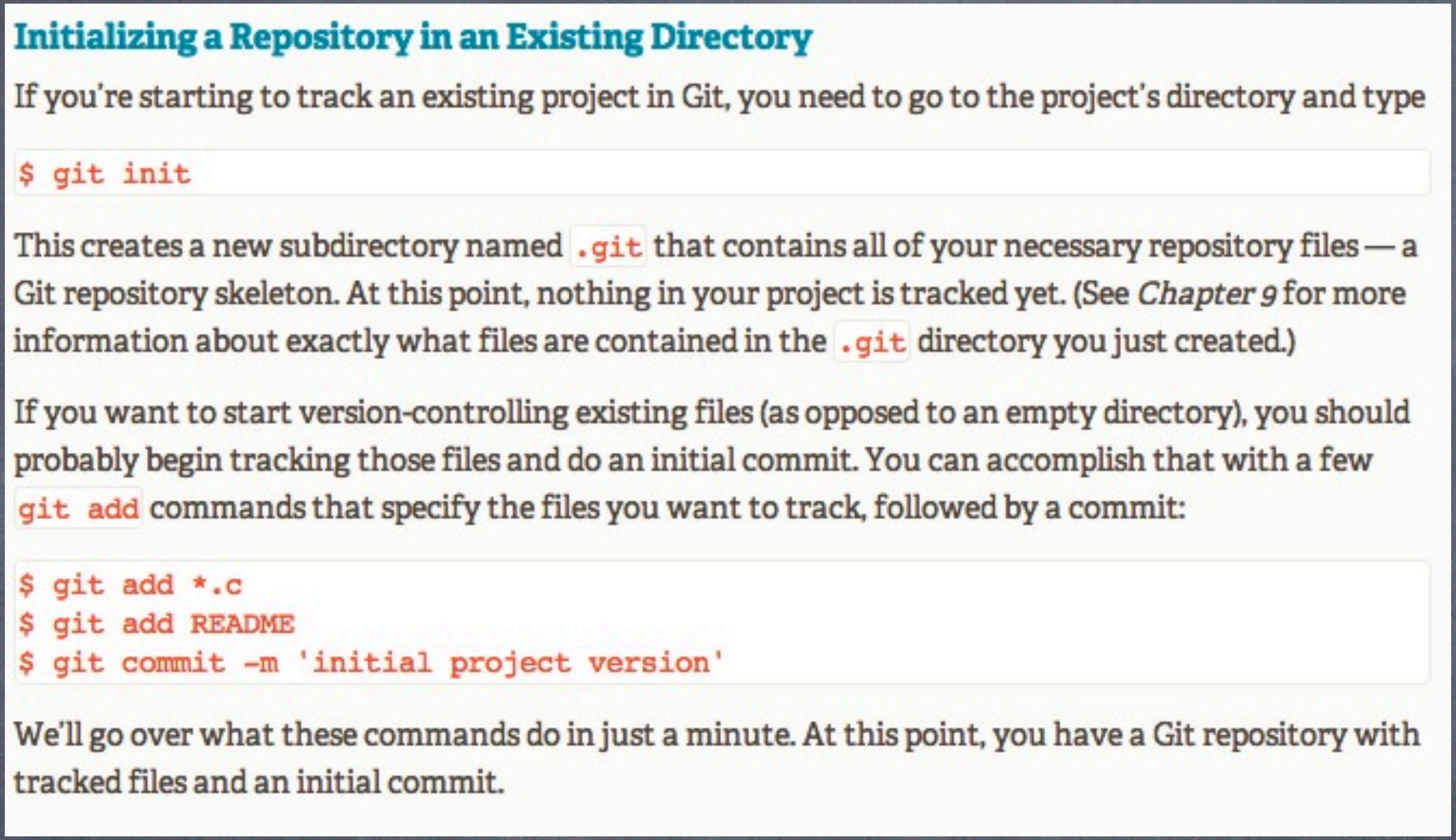
shopping cart

consistent?
“manage lists” only appears when >1 list
no move item to wish list unless logged in

functional?
can’t see count of saved item

modular
delete last item deletes list? no

a taste of git



Initializing a Repository in an Existing Directory

If you're starting to track an existing project in Git, you need to go to the project's directory and type

```
$ git init
```

This creates a new subdirectory named `.git` that contains all of your necessary repository files — a Git repository skeleton. At this point, nothing in your project is tracked yet. (See *Chapter 9* for more information about exactly what files are contained in the `.git` directory you just created.)

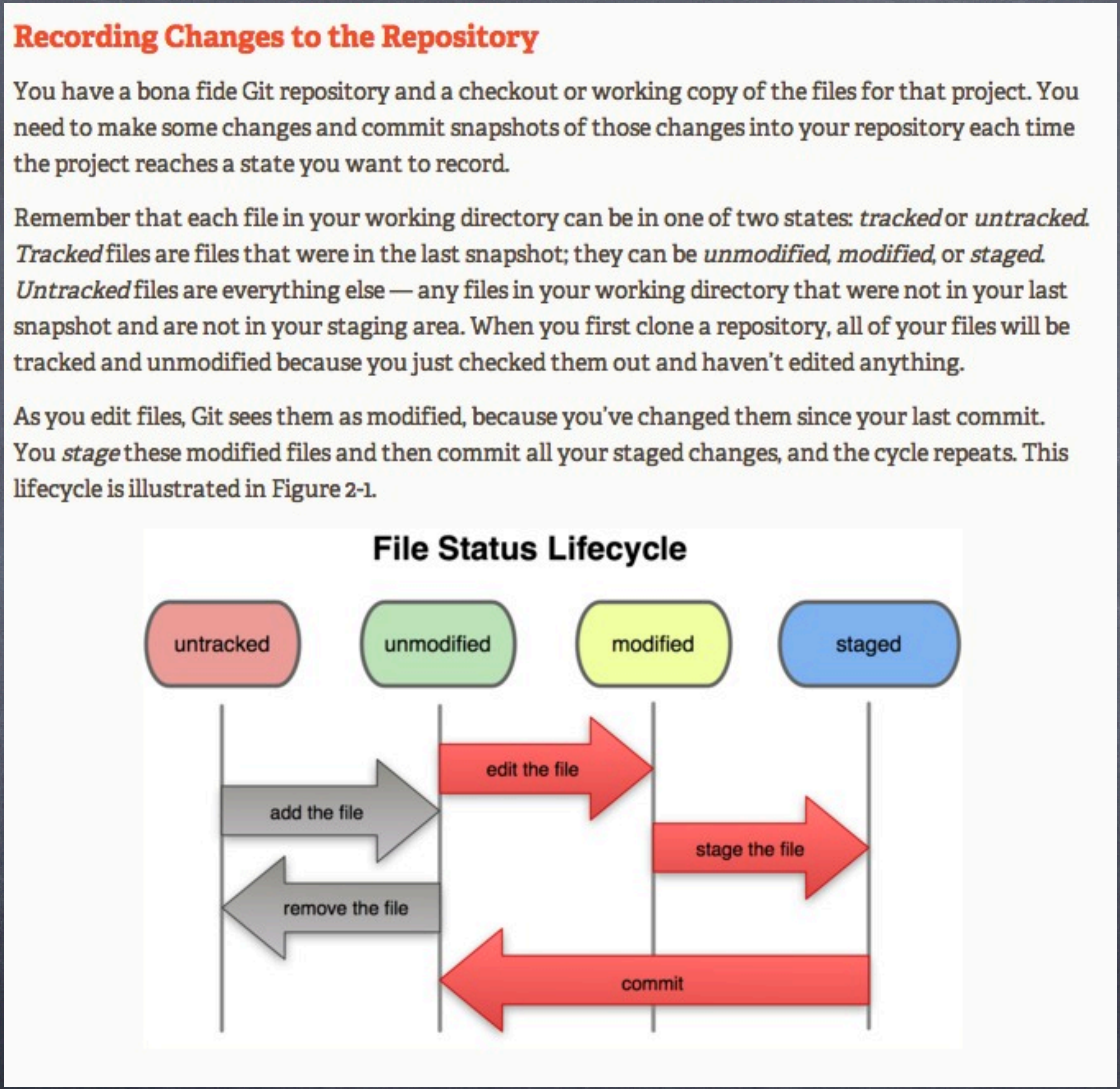
If you want to start version-controlling existing files (as opposed to an empty directory), you should probably begin tracking those files and do an initial commit. You can accomplish that with a few `git add` commands that specify the files you want to track, followed by a commit:

```
$ git add *.c
$ git add README
$ git commit -m 'initial project version'
```

We'll go over what these commands do in just a minute. At this point, you have a Git repository with tracked files and an initial commit.

sounds simple enough...

```
% git init project
Initialized empty Git repository in /Users/dnj/.../project/.git/

% cd project/
% ls

% cat >>readme.txt
This is a new project.

% git add readme.txt

% git commit -m "created readme file"
[master (root-commit) 74a2850] created readme file
 Committer: Daniel Jackson <dnj@30-86-163.dynamic.csail.mit.edu>…
 1 files changed, 1 insertions(+), 0 deletions(-)

% cd ..

% git clone project/.git/ project-copy
Cloning into project-copy...
done.

% cd project-copy
% ls
readme.txt
```

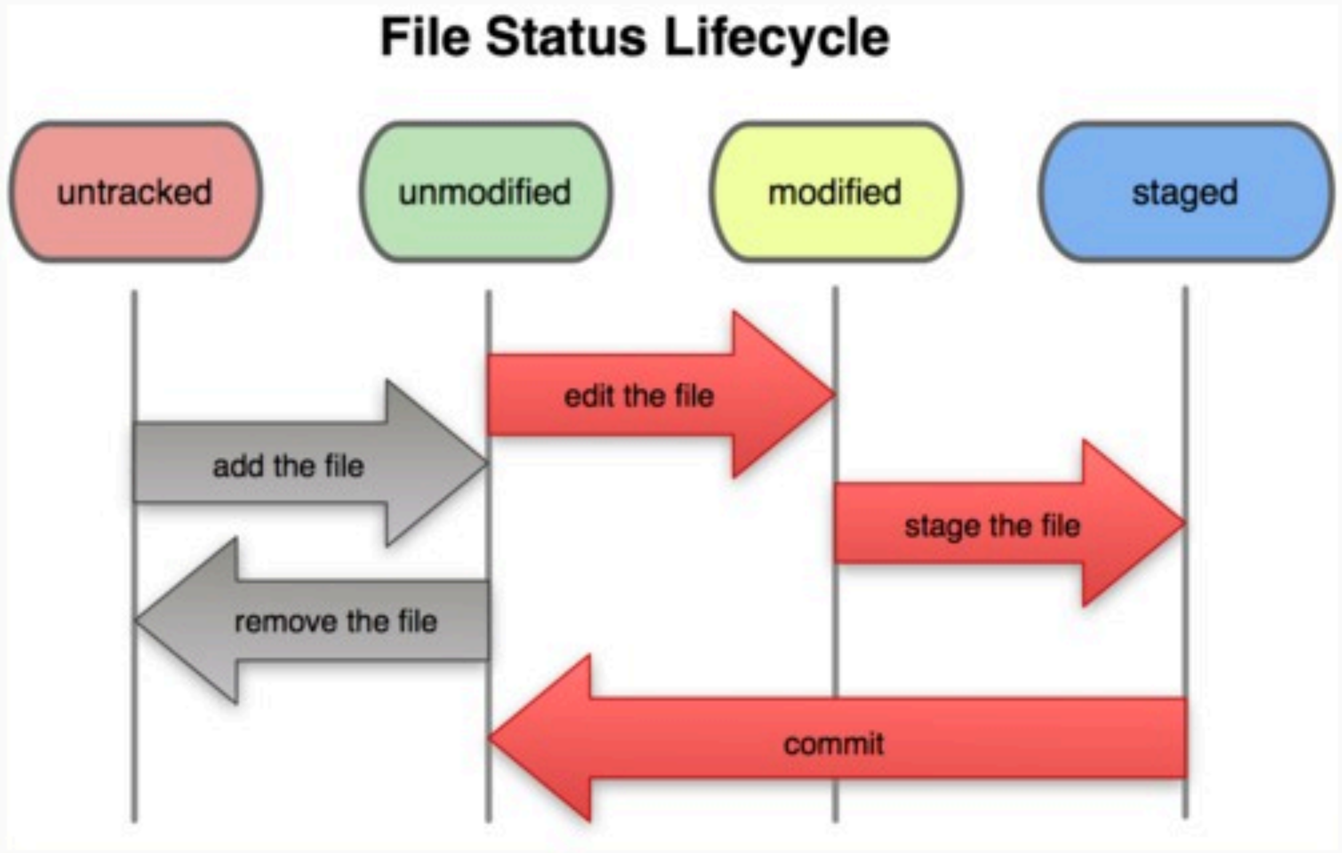


Recording Changes to the Repository

You have a bona fide Git repository and a checkout or working copy of the files for that project. You need to make some changes and commit snapshots of those changes into your repository each time the project reaches a state you want to record.

Remember that each file in your working directory can be in one of two states: *tracked* or *untracked*. *Tracked* files are files that were in the last snapshot; they can be *unmodified*, *modified*, or *staged*. *Untracked* files are everything else — any files in your working directory that were not in your last snapshot and are not in your staging area. When you first clone a repository, all of your files will be tracked and unmodified because you just checked them out and haven't edited anything.

As you edit files, Git sees them as modified, because you've changed them since your last commit. You *stage* these modified files and then commit all your staged changes, and the cycle repeats. This lifecycle is illustrated in Figure 2-1.



spoke too soon?

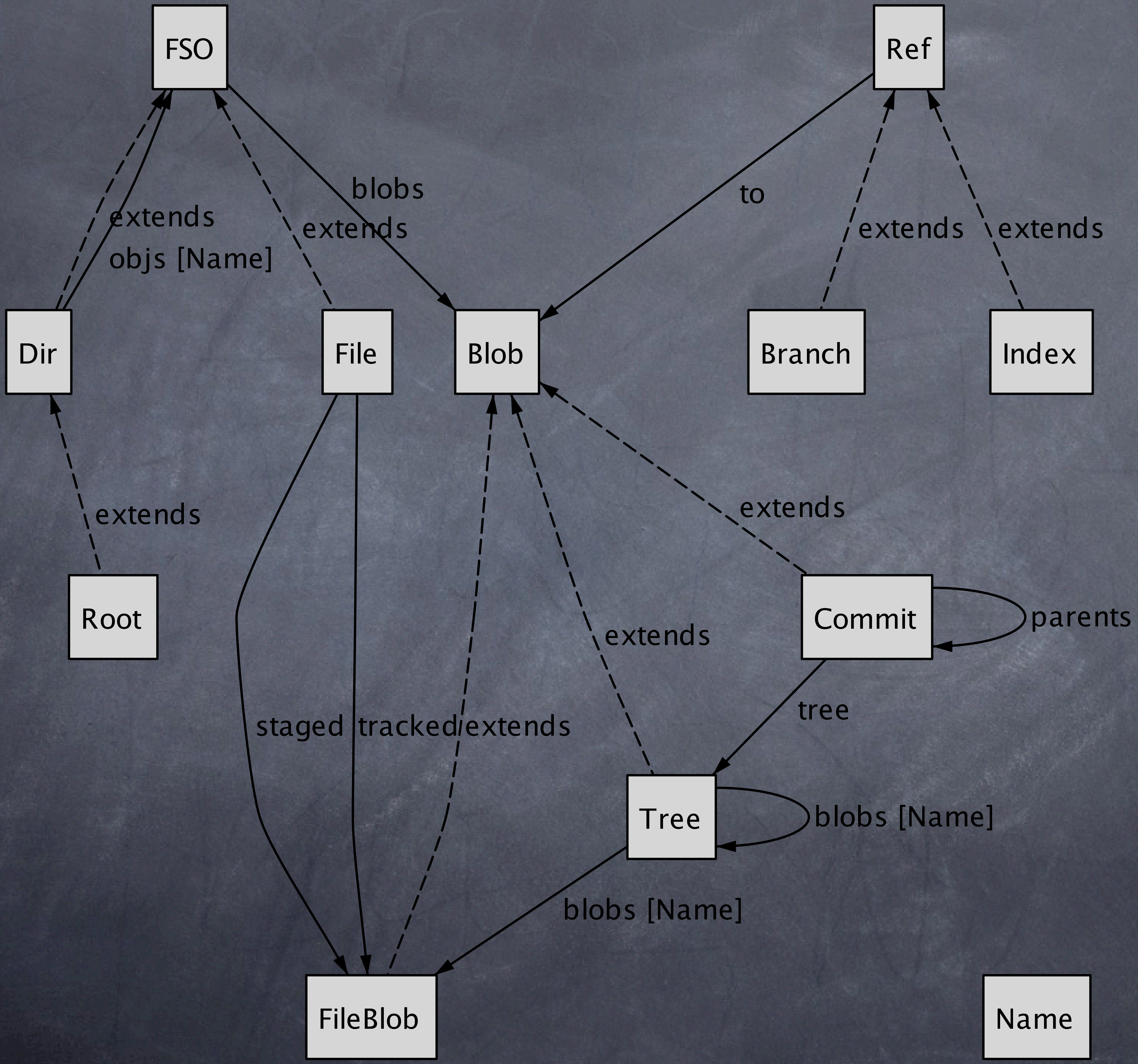
what add really does

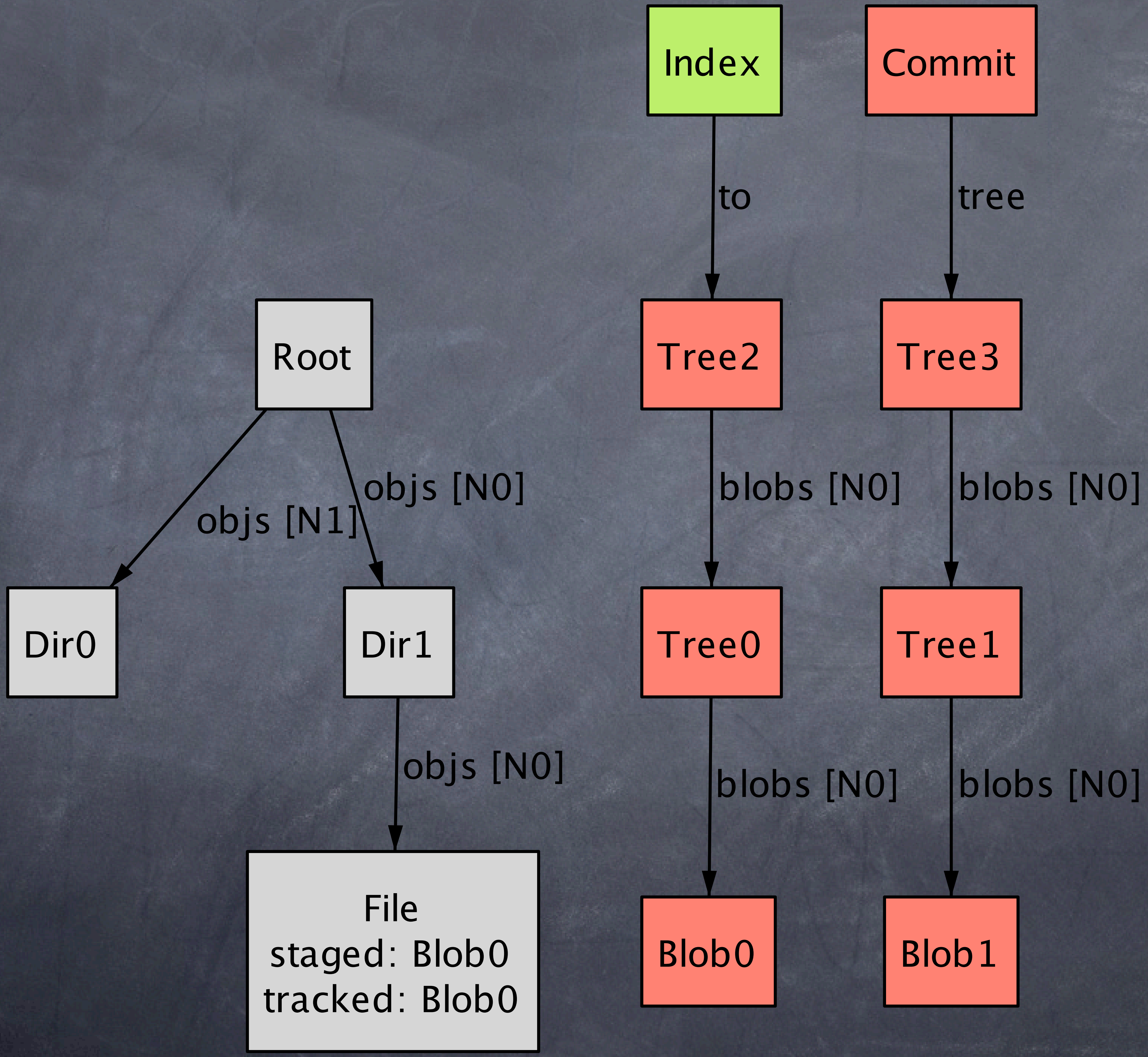
state
3 graphs: working, index, repository
index holds snapshots of files

operations
commit: copies snapshots from index to repo
add: takes snapshot of file

tracked vs. staged
“tracked”: file was once added
“staged”: latest modification added
commit -a: first add all files in index

modeling git


FSO
Ref
blobs
extends
objs [Name]
extends
to
extends
extends
Dir
File
Blob
Branch
Index
extends
extends
Root
Commit
parents
extends
tree
staged
tracked
extends
Tree
blobs [Name]
blobs [Name]
FileBlob
Name


Index
Commit
to
tree
Root
Tree2
Tree3
objs [N1]
objs [N0]
blobs [N0]
blobs [N0]
Dir0
Dir1
Tree0
Tree1
objs [N0]
blobs [N0]
blobs [N0]
File
staged: Blob0
tracked: Blob0
Blob0
Blob1

analyzing git

consistent?

same structure ⇒ same behavior

uniform

· same set, same features

directories vs. files

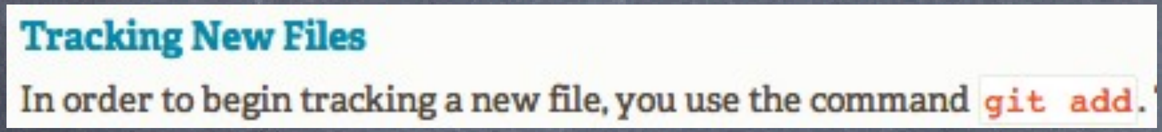
· only files are tracked
· “add d” just adds files
· can’t store empty dir

consistent?

same structure ⇒ same behavior

coherent

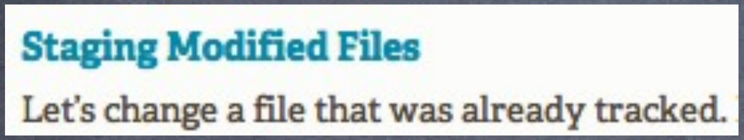
- same op, same effect



Tracking New Files

In order to begin tracking a new file, you use the command `git add`.

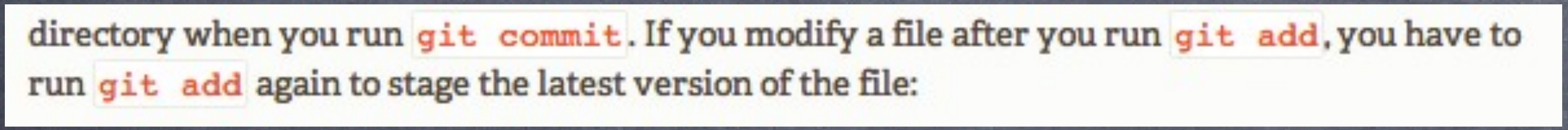
add #1: track



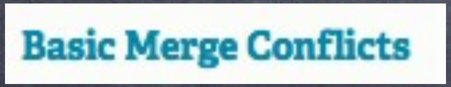
Staging Modified Files

Let's change a file that was already tracked.

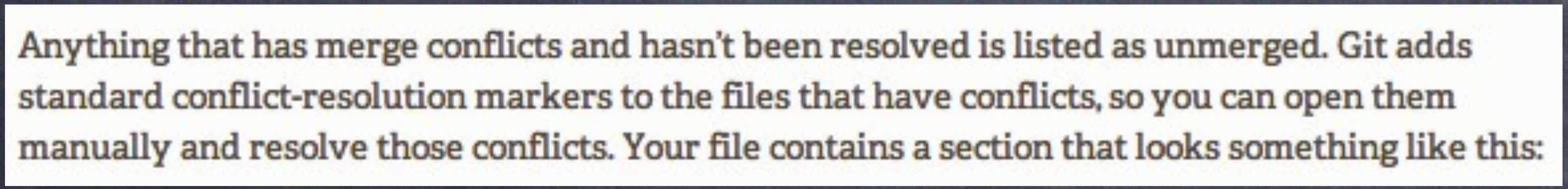
add #2: stage



directory when you run `git commit`. If you modify a file after you run `git add`, you have to run `git add` again to stage the latest version of the file:

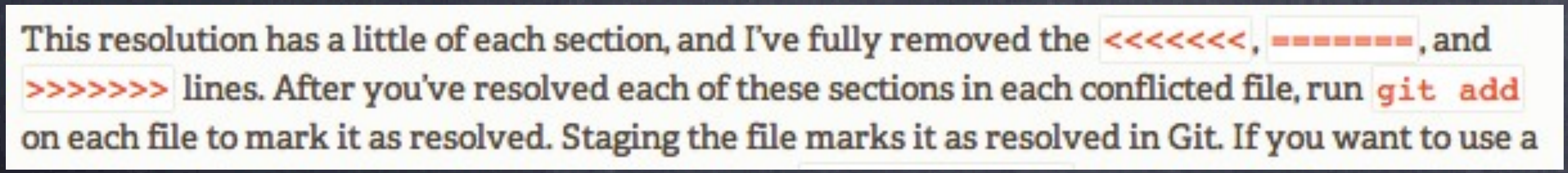


Basic Merge Conflicts



Anything that has merge conflicts and hasn't been resolved is listed as unmerged. Git adds standard conflict-resolution markers to the files that have conflicts, so you can open them manually and resolve those conflicts. Your file contains a section that looks something like this:

add #3: resolve



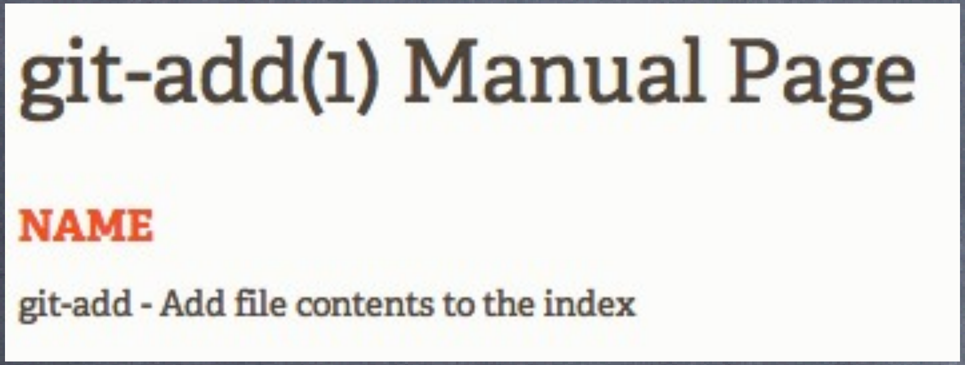
This resolution has a little of each section, and I've fully removed the `<<<<<<<`, `=======`, and `>>>>>>>` lines. After you've resolved each of these sections in each conflicted file, run `git add` on each file to mark it as resolved. Staging the file marks it as resolved in Git. If you want to use a

consistent?

same structure ⇒ same behavior

coherent

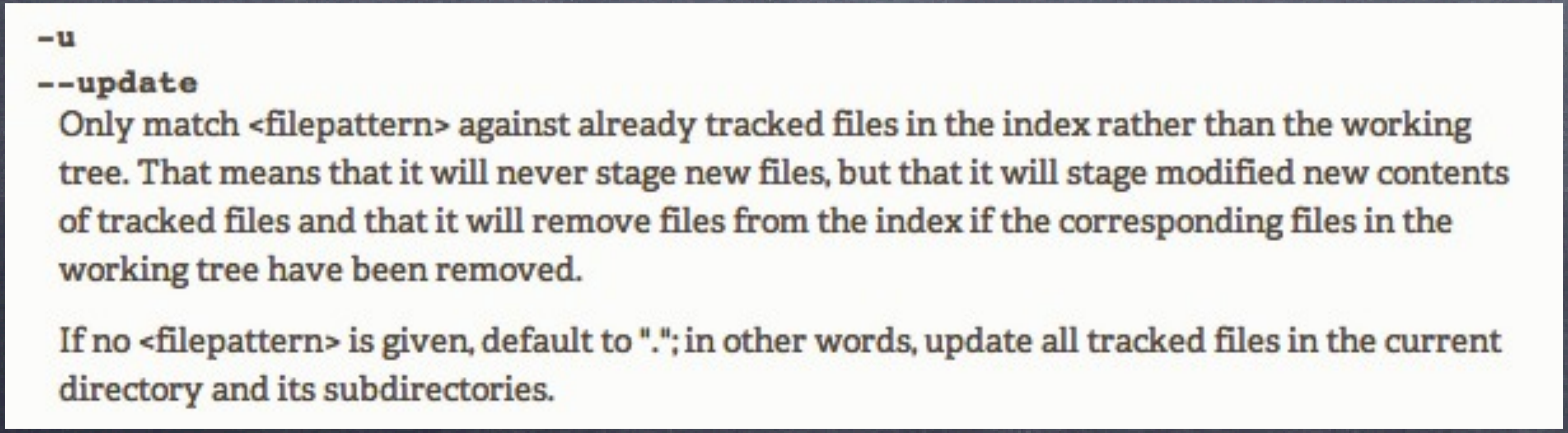
- same op, same effect



git-add(1) Manual Page

NAME

git-add - Add file contents to the index



`-u`
`--update`

Only match <filepattern> against already tracked files in the index rather than the working tree. That means that it will never stage new files, but that it will stage modified new contents of tracked files and that it will remove files from the index if the corresponding files in the working tree have been removed.

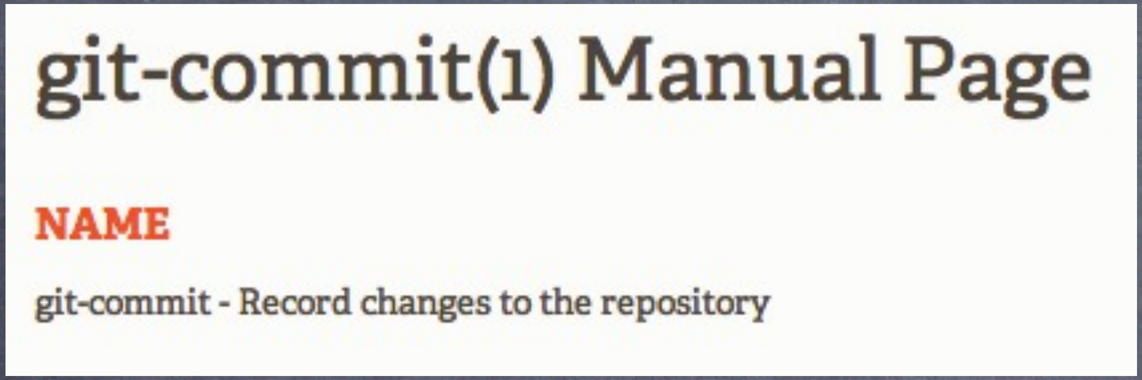
If no <filepattern> is given, default to "."; in other words, update all tracked files in the current directory and its subdirectories.

consistent?

same structure ⇒ same behavior

coherent

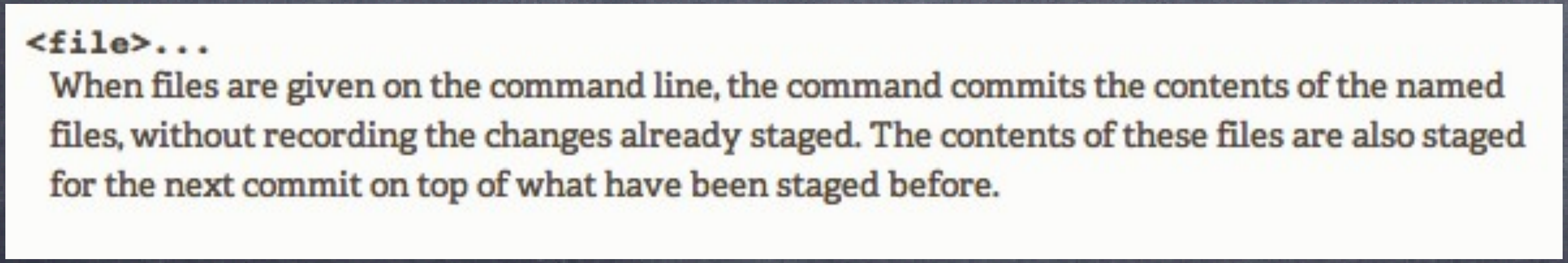
- same op, same effect



git-commit(1) Manual Page

NAME

git-commit - Record changes to the repository



```
<file>...
```

When files are given on the command line, the command commits the contents of the named files, without recording the changes already staged. The contents of these files are also staged for the next commit on top of what have been staged before.

functional?

states & operations are accessible

realizable

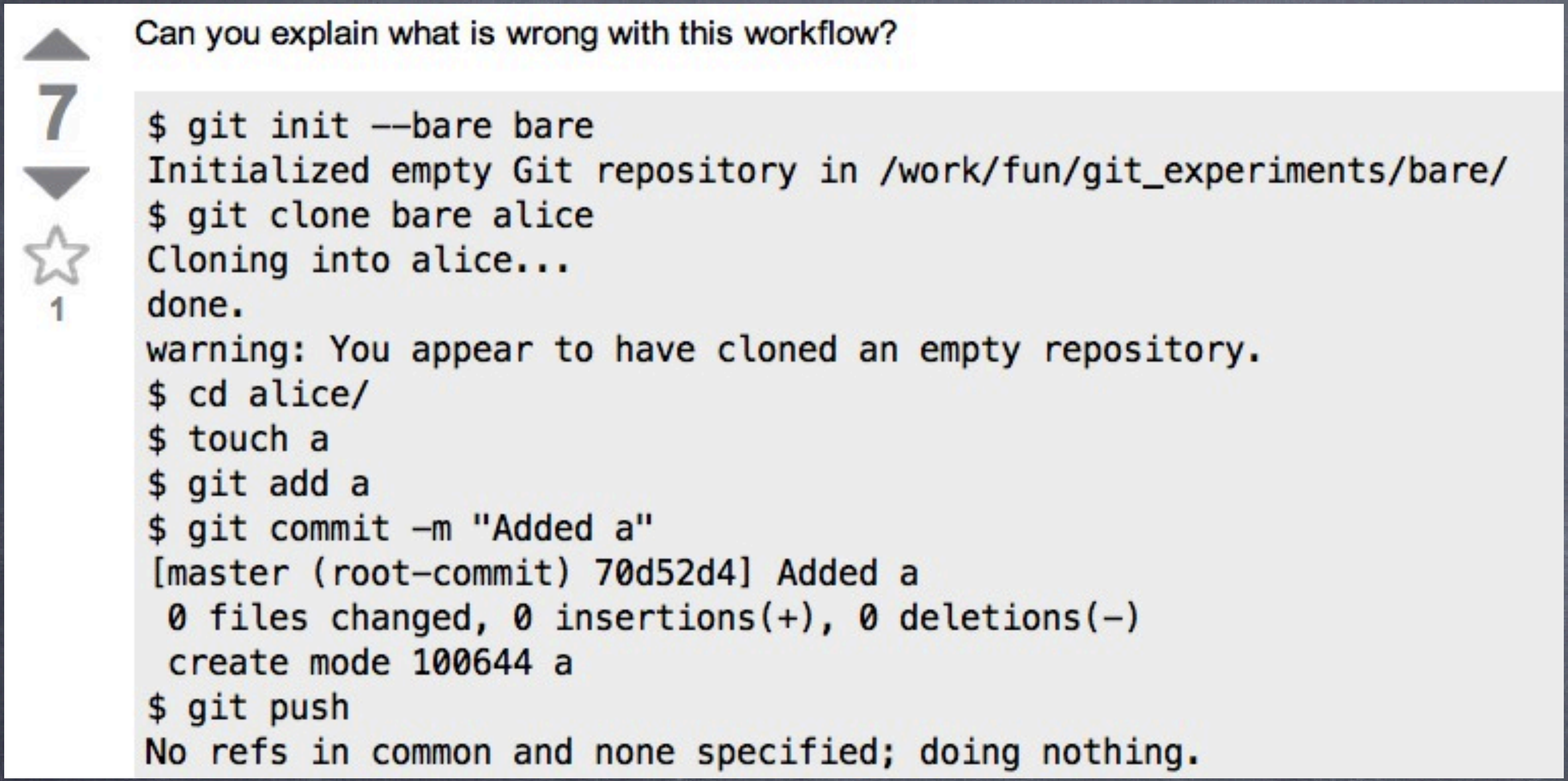
· can reach desired state
· can undo operations

can't

· undo command to track
· undo command to stage

functional?

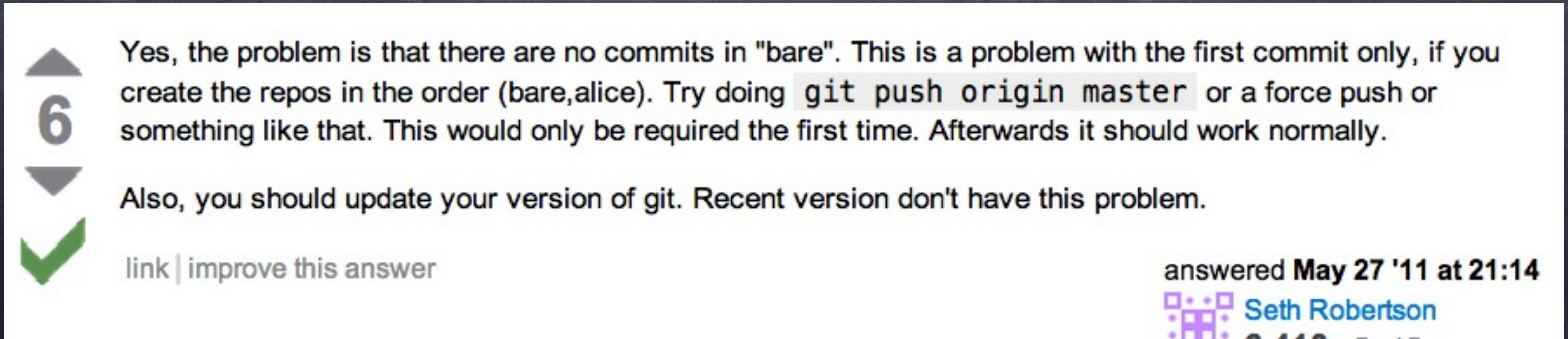
states & operations are accessible


7

1

Can you explain what is wrong with this workflow?

```
$ git init --bare bare
Initialized empty Git repository in /work/fun/git_experiments/bare/
$ git clone bare alice
Cloning into alice...
done.
warning: You appear to have cloned an empty repository.
$ cd alice/
$ touch a
$ git add a
$ git commit -m "Added a"
[master (root-commit) 70d52d4] Added a
 0 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a
$ git push
No refs in common and none specified; doing nothing.
```


6

Yes, the problem is that there are no commits in "bare". This is a problem with the first commit only, if you create the repos in the order (bare,alice). Try doing `git push origin master` or a force push or something like that. This would only be required the first time. Afterwards it should work normally.

Also, you should update your version of git. Recent version don't have this problem.

link | improve this answer

answered **May 27 '11 at 21:14**

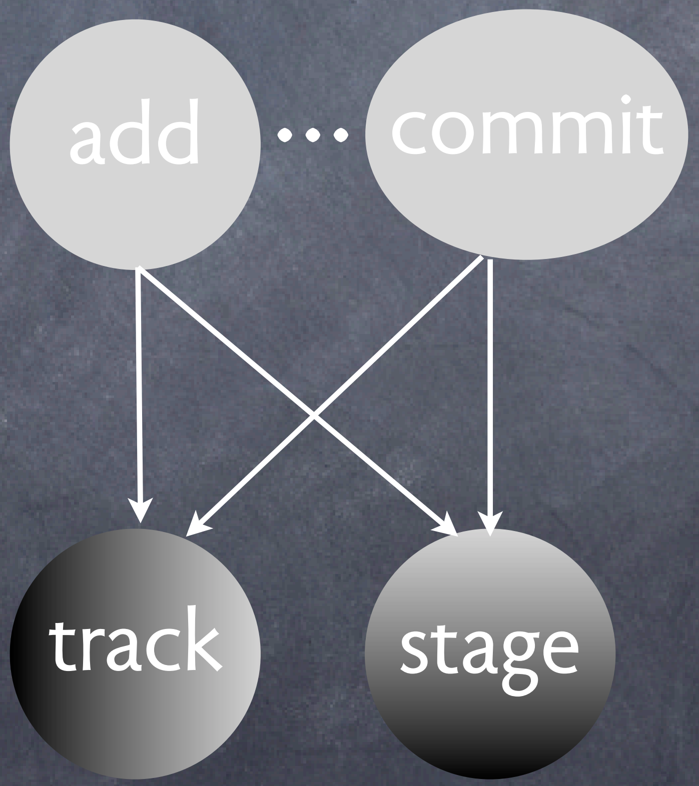
Seth Robertson

modular?

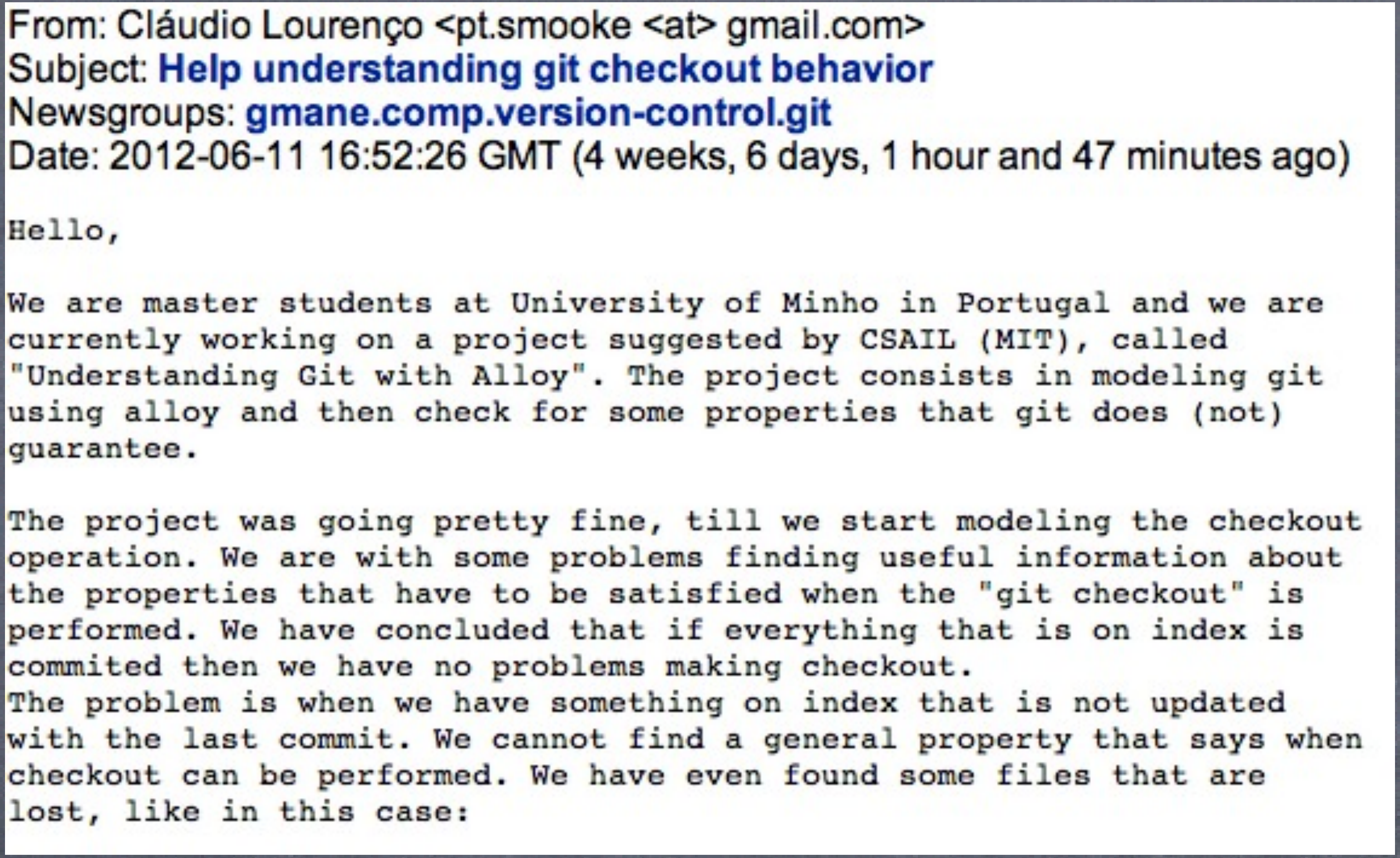
can see it and do it

decoupled

operations don't mix features

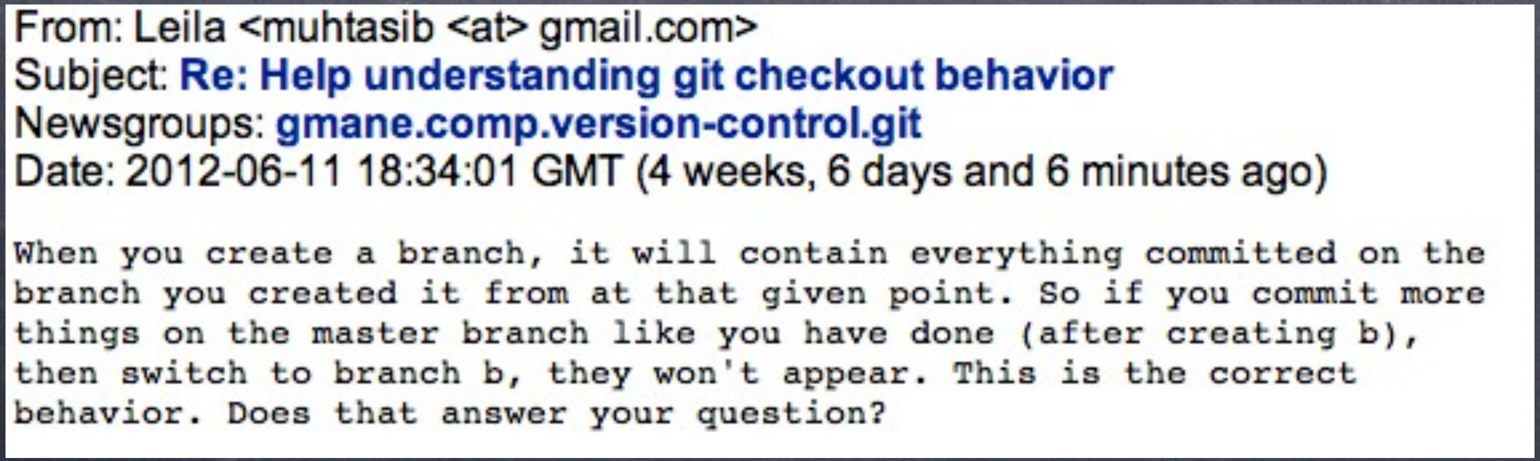


you just don't git it



From: Cláudio Lourenço <pt.smooke <at> gmail.com>
Subject: **Help understanding git checkout behavior**
Newsgroups: **gmane.comp.version-control.git**
Date: 2012-06-11 16:52:26 GMT (4 weeks, 6 days, 1 hour and 47 minutes ago)

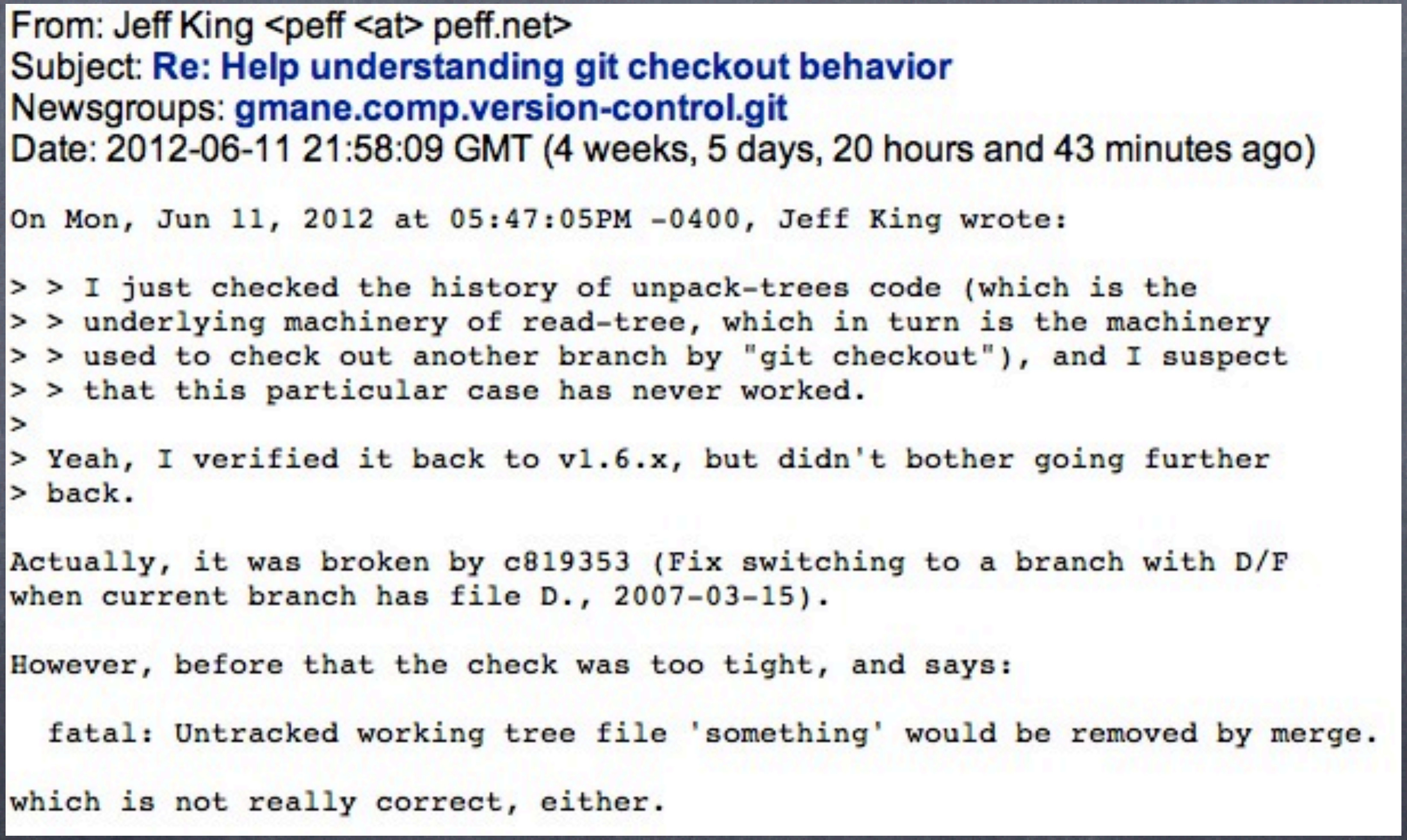
```
Hello,

We are master students at University of Minho in Portugal and we are
currently working on a project suggested by CSAIL (MIT), called
"Understanding Git with Alloy". The project consists in modeling git
using alloy and then check for some properties that git does (not)
guarantee.

The project was going pretty fine, till we start modeling the checkout
operation. We are with some problems finding useful information about
the properties that have to be satisfied when the "git checkout" is
performed. We have concluded that if everything that is on index is
commited then we have no problems making checkout.
The problem is when we have something on index that is not updated
with the last commit. We cannot find a general property that says when
checkout can be performed. We have even found some files that are
lost, like in this case:
```



From: Leila <muhtasib <at> gmail.com>
Subject: **Re: Help understanding git checkout behavior**
Newsgroups: **gmane.comp.version-control.git**
Date: 2012-06-11 18:34:01 GMT (4 weeks, 6 days and 6 minutes ago)

```
When you create a branch, it will contain everything committed on the
branch you created it from at that given point. So if you commit more
things on the master branch like you have done (after creating b),
then switch to branch b, they won't appear. This is the correct
behavior. Does that answer your question?
```

ok, it's a bug



From: Jeff King <peff <at> peff.net>
Subject: Re: Help understanding git checkout behavior
Newsgroups: gmane.comp.version-control.git
Date: 2012-06-11 21:58:09 GMT (4 weeks, 5 days, 20 hours and 43 minutes ago)

```
On Mon, Jun 11, 2012 at 05:47:05PM -0400, Jeff King wrote:

> > I just checked the history of unpack-trees code (which is the
> > underlying machinery of read-tree, which in turn is the machinery
> > used to check out another branch by "git checkout"), and I suspect
> > that this particular case has never worked.
>
> Yeah, I verified it back to v1.6.x, but didn't bother going further
> back.

Actually, it was broken by c819353 (Fix switching to a branch with D/F
when current branch has file D., 2007-03-15).

However, before that the check was too tight, and says:

  fatal: Untracked working tree file 'something' would be removed by merge.

which is not really correct, either.
```

cultural issues

worse is better

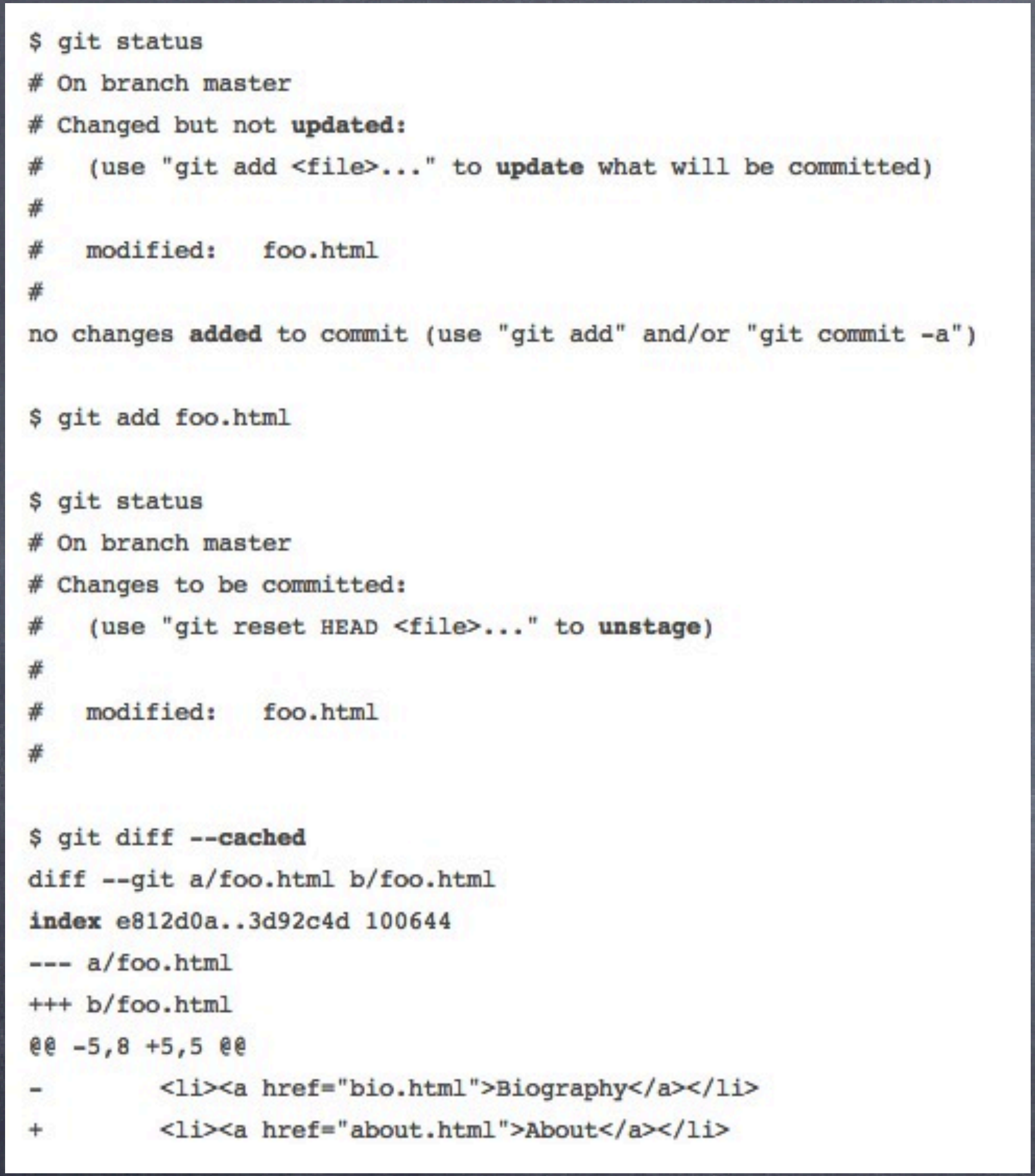
“It is more important for the implementation to be simple than the interface”

Richard Gabriel, 1991

names

A rose by any other name would smell as sweet
Shakespeare

A stage by just one name wouldn't smell as bad
Jackson



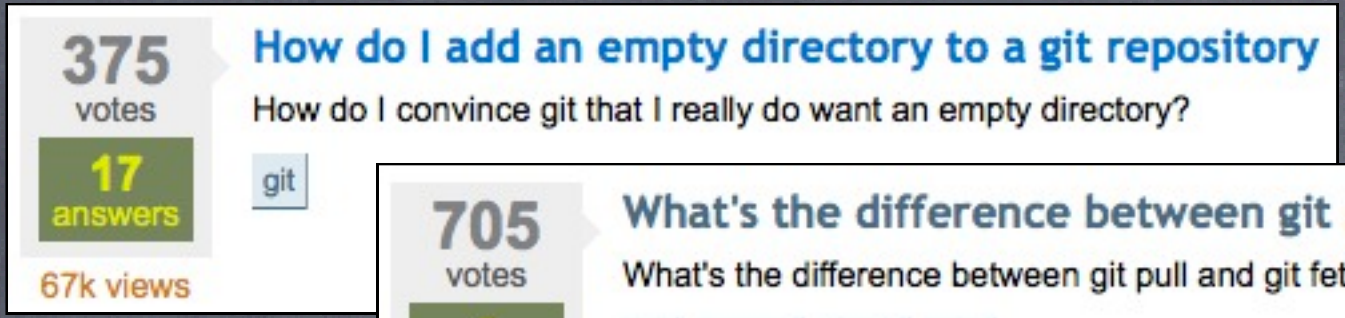
```
$ git status
# On branch master
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#
#   modified:   foo.html
#
no changes added to commit (use "git add" and/or "git commit -a")

$ git add foo.html

$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   foo.html
#

$ git diff --cached
diff --git a/foo.html b/foo.html
index e812d0a..3d92c4d 100644
--- a/foo.html
+++ b/foo.html
@@ -5,8 +5,5 @@
-         <li><a href="bio.html">Biography</a></li>
+         <li><a href="about.html">About</a></li>
```

updated
= added
= cached
= indexed
= staged

from: http://www.benspaulding.us/weblog/2009/mar/17/git-staging-call-it-what-it-is/

“Fred did that. It’s the build-up of gross pay for our weekly payroll. No-one else except Fred understands it”. His voice dropped to a reverent hush. “Fred tells me that he’s not sure he understands it himself”.

from Brilliance, Software Requirements & Specifications, Michael Jackson, Addison-Wesley 1995


375 votes

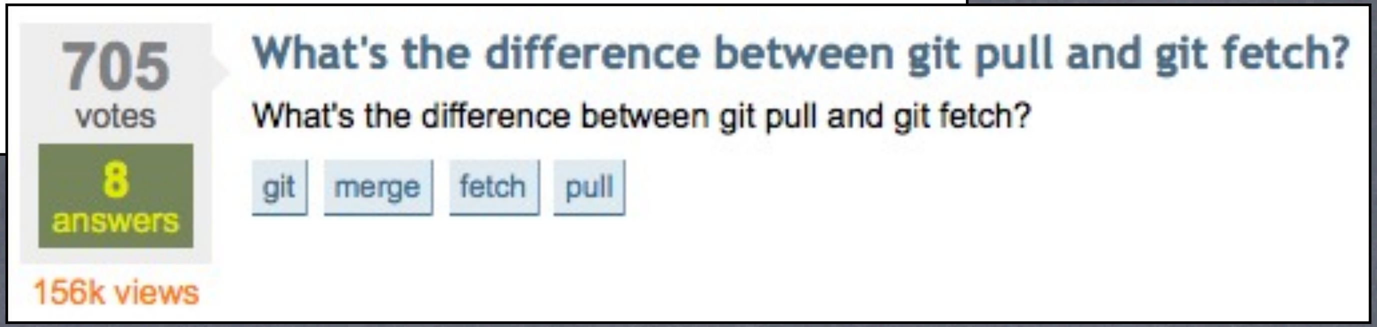
17 answers

67k views

How do I add an empty directory to a git repository

How do I convince git that I really do want an empty directory?

git


705 votes

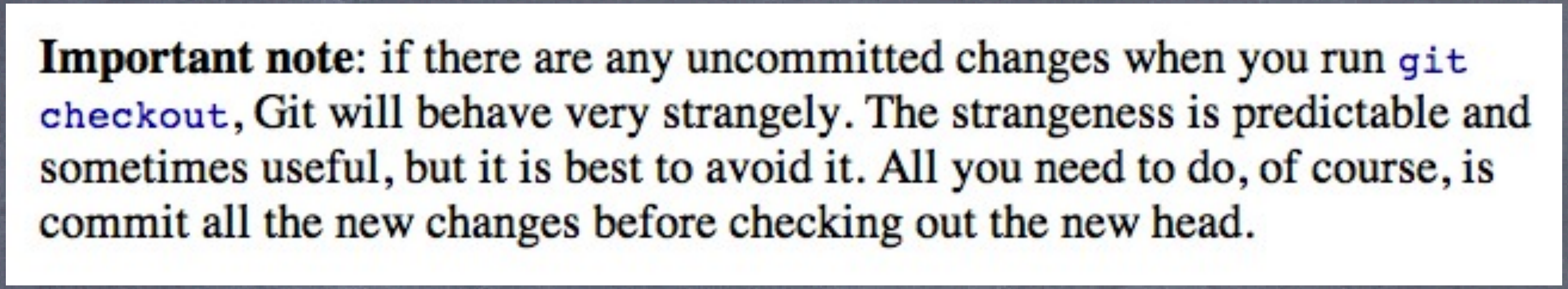
8 answers

156k views

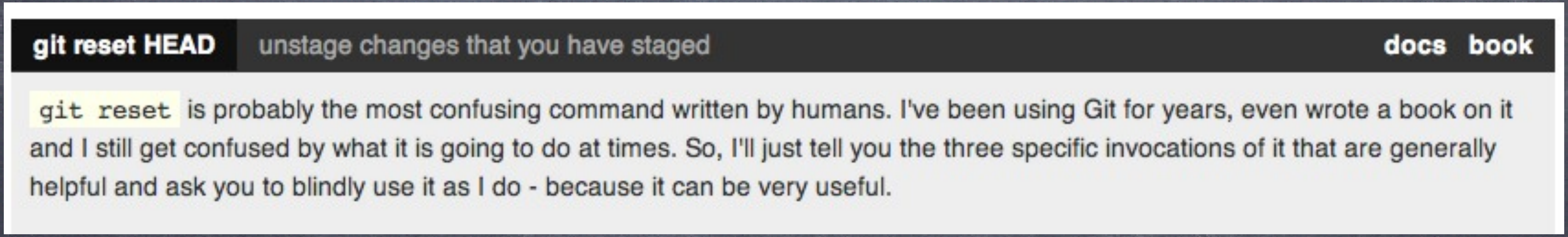
What's the difference between git pull and git fetch?

What's the difference between git pull and git fetch?

git merge fetch pull

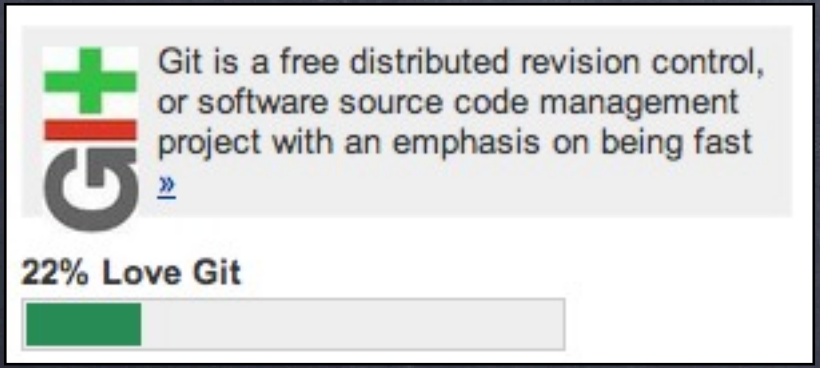

Important note: if there are any uncommitted changes when you run `git checkout`, Git will behave very strangely. The strangeness is predictable and sometimes useful, but it is best to avoid it. All you need to do, of course, is commit all the new changes before checking out the new head.

from "Understanding Git", www.sbf5.com/gcduan/technical/git/git-2.shtml

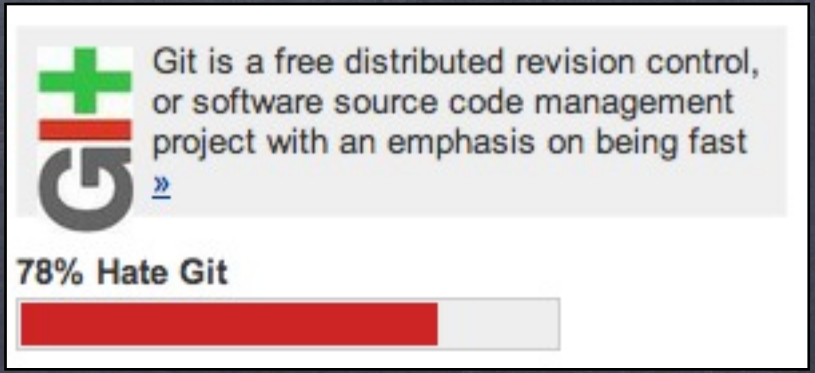

git reset HEAD unstage changes that you have staged **docs** **book**

`git reset` is probably the most confusing command written by humans. I've been using Git for years, even wrote a book on it and I still get confused by what it is going to do at times. So, I'll just tell you the three specific invocations of it that are generally helpful and ask you to blindly use it as I do - because it can be very useful.

from "Git Reference", http://gitref.org/basic/


Git is a free distributed revision control, or software source code management project with an emphasis on being fast »

22% Love Git


Git is a free distributed revision control, or software source code management project with an emphasis on being fast »

78% Hate Git

non conclusion

packaging issues

incoherence of add, commit

core issues

non-uniformity of directory vs file
coupling of staging and tracking
non-accessibility of undo operations

possible fixes

layer a new API on top (a la Eclipse)
change core: no staging?