

Surviving with Git



How to Undo Things
and Recover from Mistakes

Tobias Günther

Twitter @gntr

Email tg@fournova.com

#1

Discarding All Local Changes in a File

Apply in cases of:

“Oh my goodness! 🤔 When did I write that?!”

The screenshot shows a code editor window titled "undo-local-changes - Working Copy (master - 4 Changed Files)". The editor displays the file "imprint.html" with the following content:

```
@@ -1,11 +1,11 @@
1 1 <html>
2 2
3 3 <head>
-4 4 <title>Tower of Imprint</title>
+4 4 <title>Looks like a stupid change...</title>
5 5 <link rel="image/x-icon" href="img/favicon.ico" type="image/x-icon" />
6 6 <link type="text/css" rel="stylesheet" href="css/general.css"/>
7 7 </head>
8 8
9 9
10 10
11 11 <body>
```

A yellow arrow points from the text "that?!" in the title above to the new title text in the code. A yellow circle highlights the new title text: "<title>Looks like a stupid change...</title>".

#1

Discarding All Local Changes in a File

```
$ git checkout HEAD <filename>
```

Discarding uncommitted local changes cannot be undone!

#2

Restoring a Deleted File

Apply in cases of:

“Heavens above! 🤔 Why did I delete that?!?”

The screenshot shows a code editor interface with a dark theme. The top bar indicates the current repository is 'carparts-website (master)' and the working copy is active. The file list on the left shows three files: 'error.html' (deleted, marked with a red 'D'), 'img/icon.png' (added, marked with a green 'A'), and 'index.html' (modified, marked with a blue 'M'). The 'error.html' file is highlighted with a yellow circle. The editor pane shows the content of 'error.html', which is a deleted file with mode 100644 (Regular). The content is HTML code, starting with a head section containing a title 'Tower :: Imprint', a shortcut icon link, and a stylesheet link. The body section contains a content wrapper and a navigation element. A yellow arrow points from the text 'Heavens above! 🤔 Why did I delete that?!?' to the 'error.html' file in the file list.

```
Deleted file with mode 100644 (Regular)
1 chunk, 0 insertions, 43 deletions

@@ -1,43 +0,0 @@
-1      0  <html>
-2
-3      <head>
-4          <title>Tower :: Imprint</title>
-5          <link rel="shortcut icon" href="img/favicon.ico"
-6              type="image/x-icon" />
-7          <link type="text/css" rel="stylesheet" href="css/
-8              general.css"/>
-9      </head>
-10
-11     <body>
-12         <div id="contentWrapper">
-13             <div id="navigation">
```

#2

Restoring a Deleted File

```
$ git checkout HEAD <filename>
```

#3 Discarding Chunks / Lines in a File

Apply in cases of:

“Mama mia! 🙄 Was that me?!!”

Repositories > carparts-website_orig (master) > Working Copy Clone

index.html Staged Unstaged

1 chunk, 3 insertions, 2 deletions

@@ -13,7 +13,8 @@ Discard Lines Stage Lines

Line	Old	New	Content
13	13		<div id="navigation">
14	14		
15	15		Home
		-16	About
		+16	About Us
		+17	Product
17	18		Imprint
18	19		
19	20		</div>

@@ -27,7 +28,7 @@ Discard Chunk Stage Chunk

Line	Old	New	Content
27	28		section.
28	29		</p>
29	30		<p class="blogParagraph">
		-30	Read our blog.
		+31	Read our new blog.

#3 Discarding Chunks / Lines in a File

```
$ git checkout -p <filename>
```

#4 Discarding All Local Changes

Apply in cases of:

“Bloody hell! 🙊 Let’s start over!”

The screenshot shows a code editor interface with a diff view for the file `index.html`. The interface includes a sidebar on the left with a file list, a central diff view, and a right sidebar with commit options.

File List (Left Sidebar):

Status	Filename
<input type="checkbox"/>	M index.html
<input type="checkbox"/>	? img/icon.png
<input type="checkbox"/>	? product.html
<input type="checkbox"/>	M about.html
<input type="checkbox"/>	D error.html

Diff View (Center):

@@ -13,7 +13,8 @@

Line	Line	Code
13	13	<div id="navigation">
14	14	
15	15	Home
-16	+16	About
	+16	About Us
	+17	Product
17	18	Imprint
18	19	
19	20	</div>

@@ -27,7 +28,7 @@

Line	Line	Code
27	28	Visit our "learn"
28	29	section.
28	29	</p>
29	30	<p class="blogParagraph">
-30	+31	Read our blog.
	+31	Read our new blog.

#4

Discarding All Local Changes

```
$ git reset --hard HEAD
```

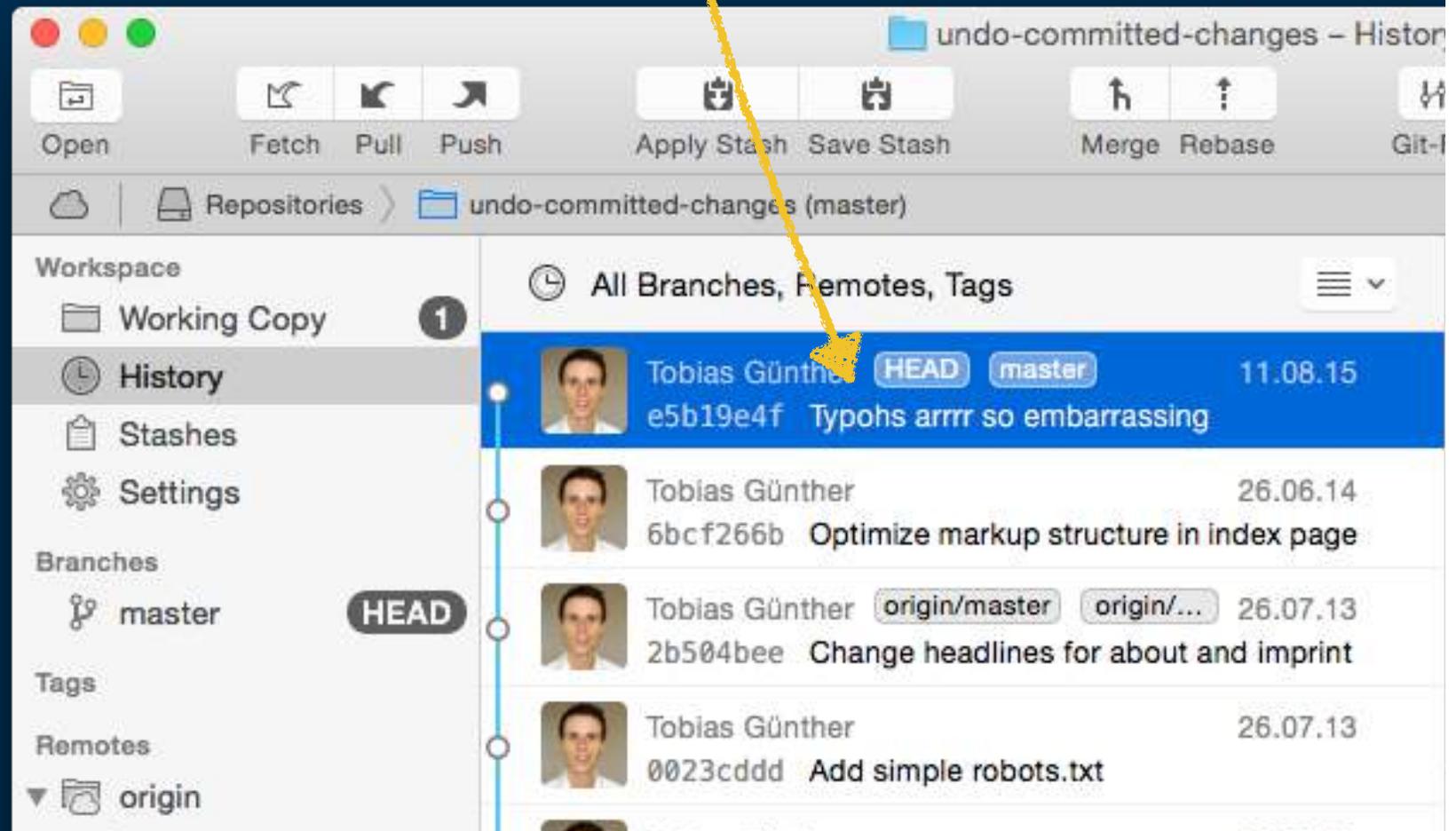
Please be careful with this command:
discarding uncommitted local changes cannot be undone!

#5

Fixing the Last Commit

Apply in cases of:

“Holy moly! 🤔 Too quick with that commit...”



#5

Fixing the Last Commit

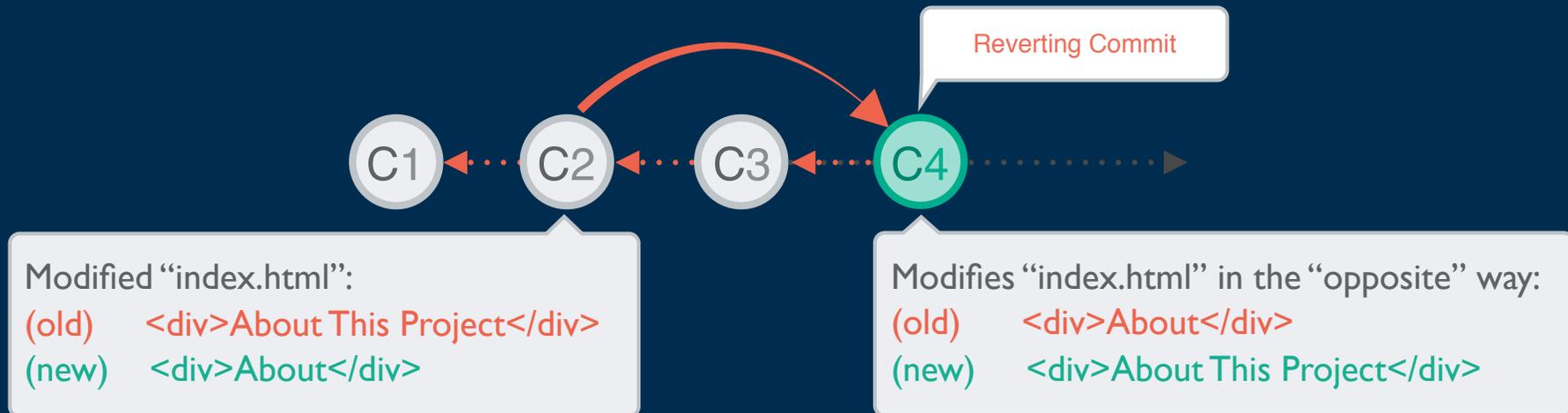
```
$ git commit --amend -m "correct"
```

“--amend” rewrites history! Never change history for commits that have already been pushed to a remote repository!

#6 Reverting a Commit in the Middle

Apply in cases of:

“Deary me! 🙄 How can I get rid of that commit?”



“git revert” creates a new commit (C4) that reverts the effects of a specified commit (C2).

#6 Reverting a Commit in the Middle

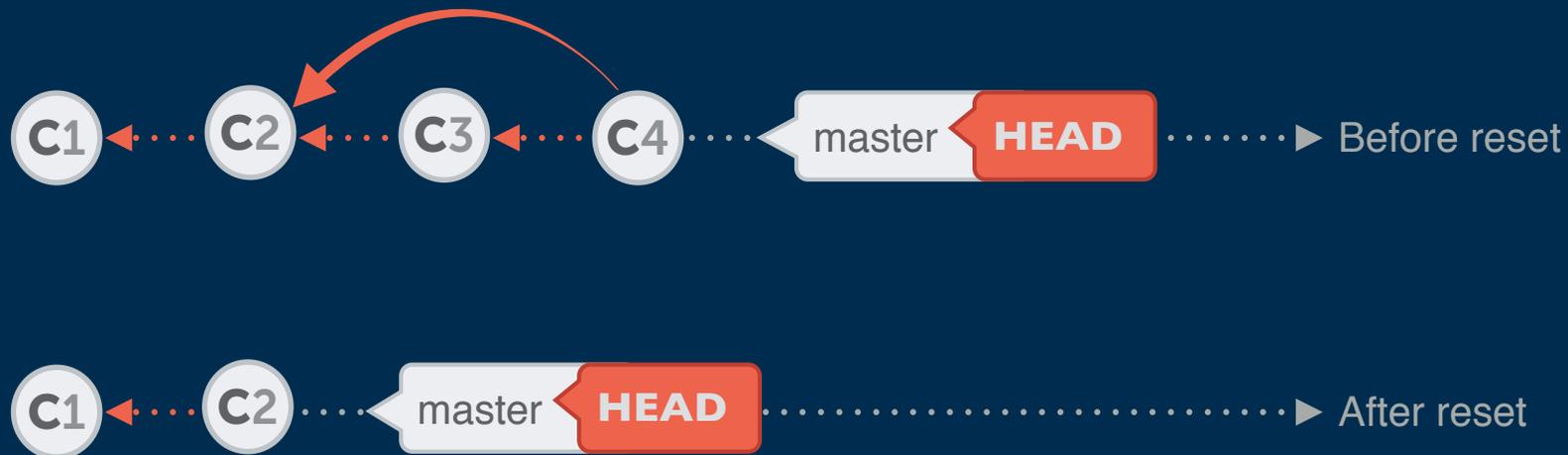
```
$ git revert 2b504bee
```

#7

Resetting to an Old Revision

Apply in cases of:

“Good gravy! 😞 I wish I could turn back time...”



“git reset” sets your HEAD pointer (and thereby also your working copy) to an older revision. Commits that came after this revision appear to be undone.

#7

Resetting to an Old Revision

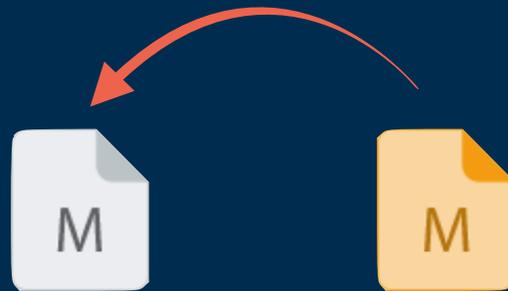
```
$ git reset --hard 0023cdd  
$ git reset --mixed 0023cdd
```

#8

Resetting a File to an Old Revision

Apply in cases of:

“Yikes! 🤔 I know this file was correct sometime!”



#8 Resetting a File to an Old Revision

```
$ git checkout 6bcf266b -- index.html
```

Reflog



a journal that logs every movement of the HEAD pointer

```
Mac-a0999b076373:reflog tobidobi$ git reflog
0023cdd HEAD@{0}: reset: moving to 0023cdddf42d916bd7e3d0a279c1f36bfc8a051b
776f8ca HEAD@{1}: checkout: moving from development to master
2aea65a HEAD@{2}: commit: New about and error page titles
6bcf266 HEAD@{3}: checkout: moving from master to development
776f8ca HEAD@{4}: commit: Change about title and delete error page
6bcf266 HEAD@{5}: reset: moving to 6bcf266b78872073d750c94809fe73d21ac7934a
798d1b6 HEAD@{6}: commit: Change about page
6bcf266 HEAD@{7}: checkout: moving from develop to master
6bcf266 HEAD@{8}: checkout: moving from master to develop
6bcf266 HEAD@{9}: commit (amend): Optimize markup structure in index page
```

#9 Recovering Deleted Commits

Apply in cases of:

“Sweet baby Jesus! 🤯 Did I just delete the wrong commits?!”

1. You think you want to reset / roll back.



2. You reset.



3. You notice it was a bad idea.

4. You panic.



#9

Recovering Deleted Commits

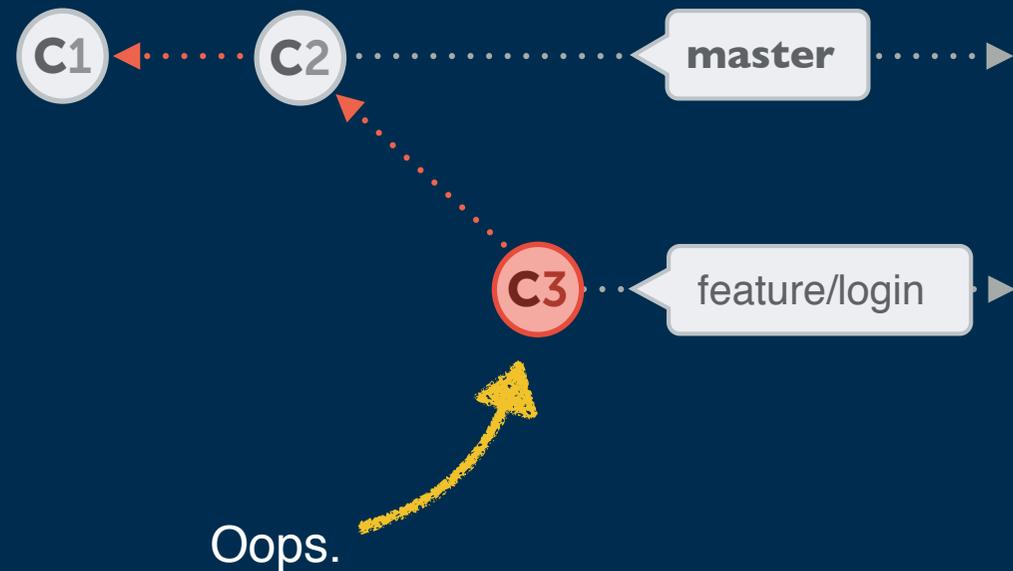
```
$ git reflog  
$ git branch happy-ending e5b19e4
```

#10 Recovering a Deleted Branch

Apply in cases of:

“God Almighty! 🤯 Did I just delete the wrong branch?!!”

1. You think you don't need a feature branch anymore.
2. You delete it.
3. You notice it was a bad idea.
4. You panic. 🤯



#10

Recovering a Deleted Branch

```
$ git reflog
```

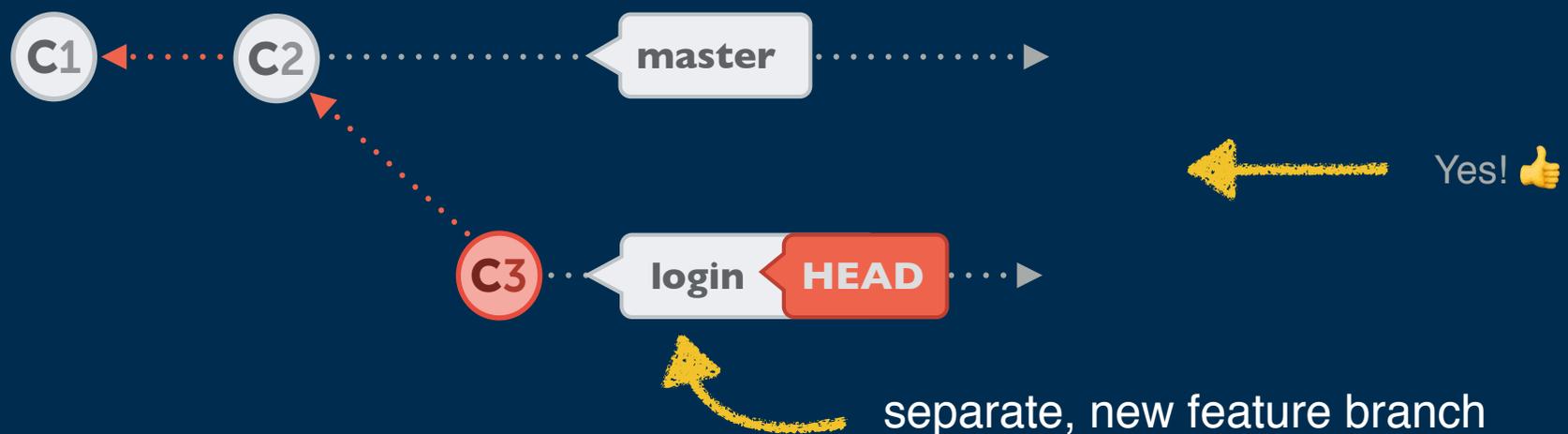
```
$ git branch develop b1c249b
```

#11

Moving a Commit to a New Branch

Apply in cases of:

“Damn it! 😞 I should have created a new branch before committing!”



#11 Moving a Commit to a New Branch

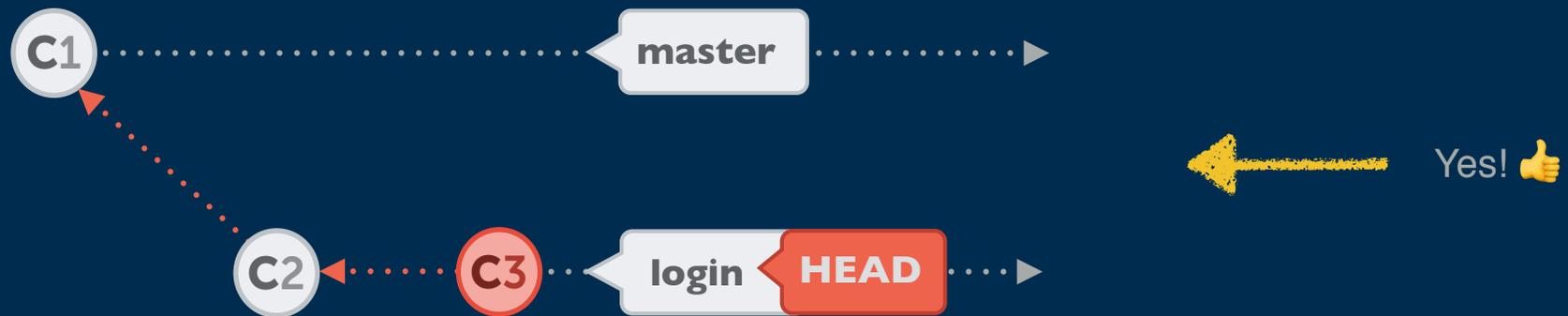
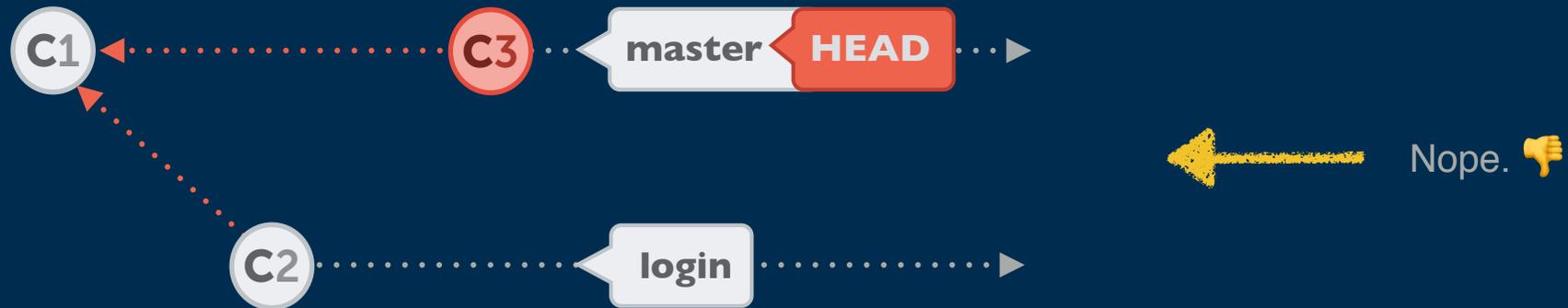
```
$ git branch feature/login  
$ git reset HEAD~1 --hard
```

#12

Moving a Commit to a Different Branch

Apply in cases of:

“Ay, caramba! 🤔 I should have committed this to a different branch!”



that's where the commit actually belonged!

#12 Moving a Commit to a Different Branch

```
$ git checkout feature/newsletter  
$ git cherry-pick <SHA>  
$ git checkout master  
$ git reset --hard HEAD~1
```

Interactive Rebase

the Swiss Army knife of Git tools

(please don't cut yourself!)

```
git-rebase-todo — rebase-merge (git: master)
1 pick 0023cdd Add simple robots.txt
2 pick 2b504be Change headlines for about and imprint
3 pick 6bcf266 Optimize markup structure in index page
4
5 # Rebase 113adcc..6bcf266 onto 113adcc (3 commands)
6 #
7 # Commands:
8 # p, pick = use commit
9 # r, reword = use commit, but edit the commit message
10 # e, edit = use commit, but stop for amending
11 # s, squash = use commit, but meld into previous commit
12 # f, fixup = like "squash", but discard this commit's log message
```

#13

Editing Old Commit Messages

Apply in cases of:

“Christ on a cracker! 🤪 Did I write this commit message?!”

“Dear coworkers. Here is my
PayPal password. Have fun.”



Interactive Rebase Step by Step

1 How far back do you want to go? What should be the "base" commit?

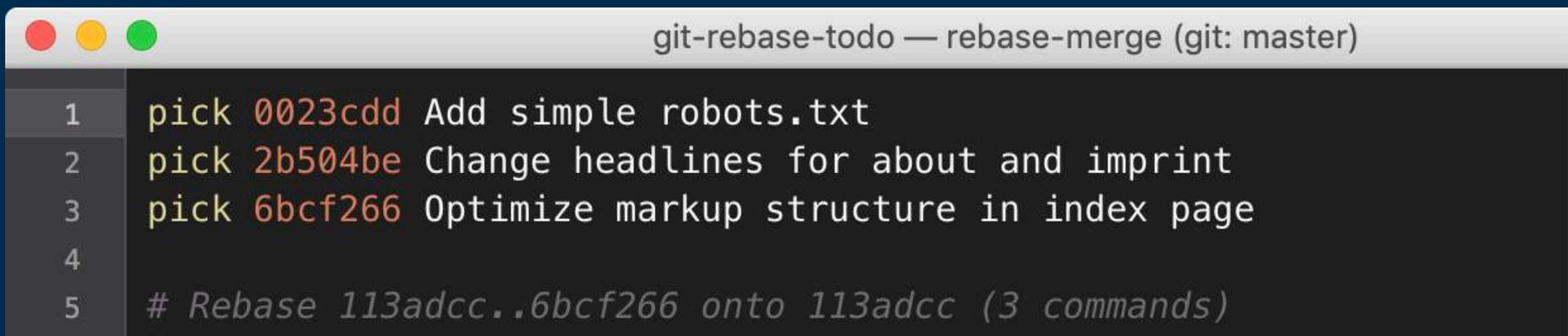
2

```
$ git rebase -i HEAD~3
```

3 In the editor, only determine which actions you want to perform.

Don't change commit data in this step, yet!

Notice: Commits are in "reverse" order!



```
git-rebase-todo — rebase-merge (git: master)
1 pick 0023cdd Add simple robots.txt
2 pick 2b504be Change headlines for about and imprint
3 pick 6bcf266 Optimize markup structure in index page
4
5 # Rebase 113adcc..6bcf266 onto 113adcc (3 commands)
```

#13

Editing Old Commit Messages

```
$ git rebase -i HEAD~3
```

...then use “reword” option

#14

Deleting Old Commits

Apply in cases of:

“Knock me over with a feather! 🤔 Did I create that commit?!”



#14 Deleting Old Commits

```
$ git rebase -i HEAD~3
```

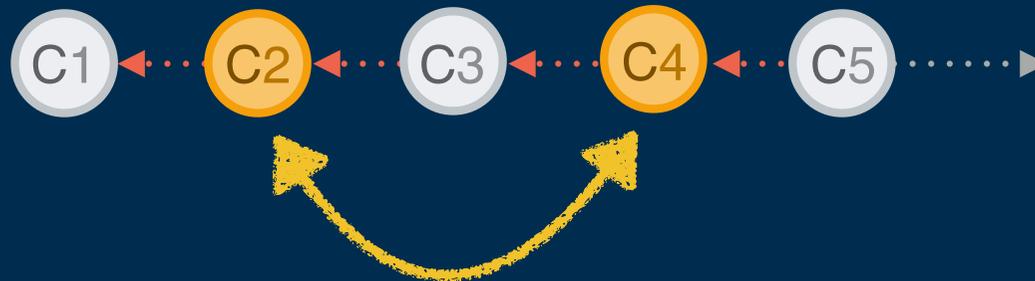
...and remove lines for unwanted commits

#15

Squashing Multiple Commits Into One

Apply in cases of:

“Holy cow! 🐮 Why did I create so many commits??”



let's combine these
two into one

#15

Squashing Multiple Commits Into One

```
$ git rebase -i HEAD~3
```

...then use “squash” option

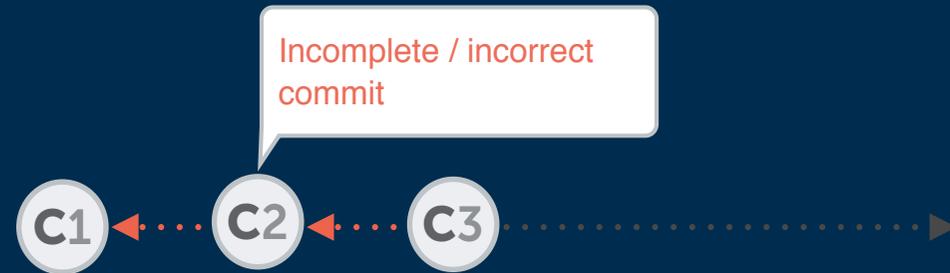
#16

Adding a Change to an Old Commit

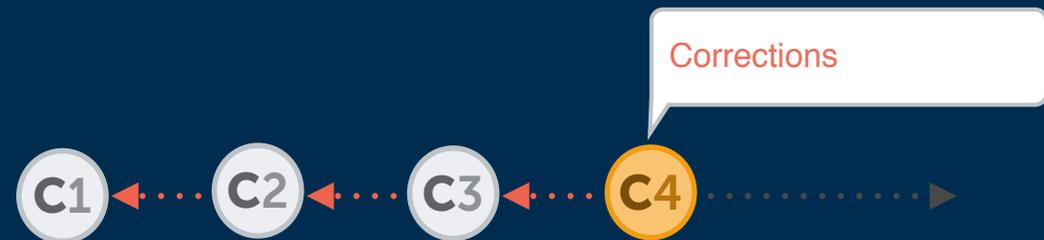
Apply in cases of:

“Jiminy Cricket! 🙄 I forgot to add a change!”

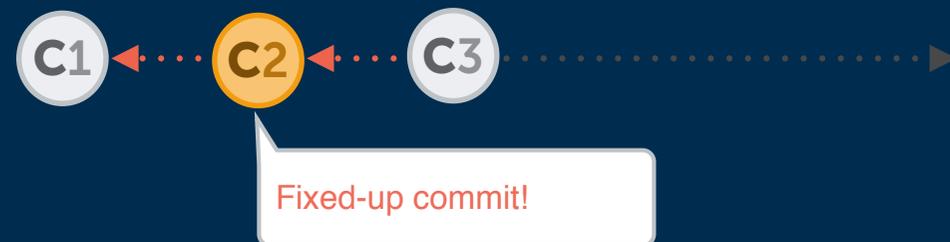
1. You forgot to commit something - or simply made a mistake!



2. You add your corrections in a normal commit - but with --fixup



3. Your changes are added to the original commit.



#16

Adding a Change to an Old Commit

```
$ git commit --fixup 2b504bee
```

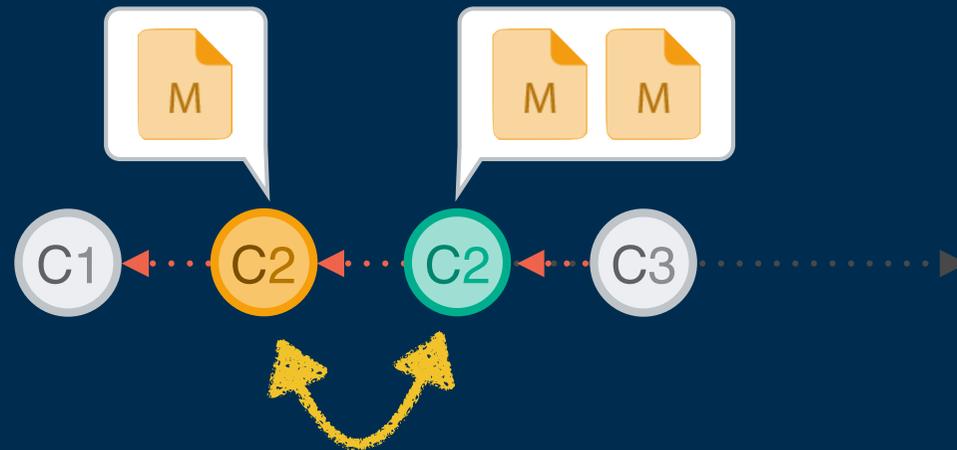
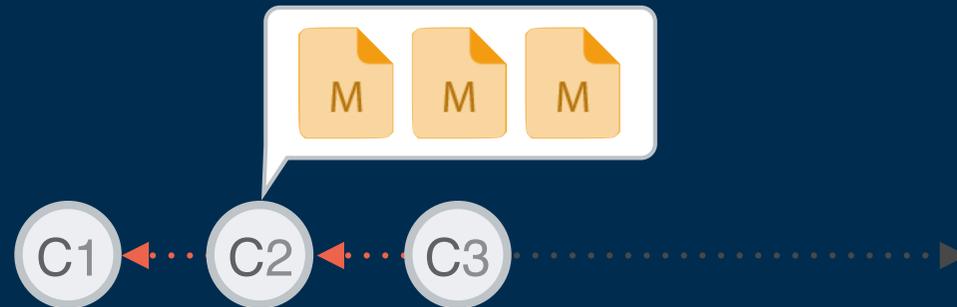
```
$ git rebase -i --autosquash HEAD~4
```

#17

Splitting / Editing an Old Commit

Apply in cases of:

“Great Ceasar! 🤔 Can I retry this commit once more?”



split one commit into
multiple new commits

Attention!

Caution When Rewriting History!

Do NOT rewrite commit history that has already been pushed to a remote repository!

```
$ git commit --amend  
$ git rebase  
$ git rebase -i  
$ git reset  
...
```

use with care



Free Videos!

To Help You Remember What You Learned...

www.git-tower.com/learn/git/first-aid-kit

17 Videos



1 Cheat Sheet

First Aid Kit for Git

Postcards!

A Tiny Little Gift...

put it on your desk

send it to a friend



CODING IS AN ART. LEARN FROM OTHERS. HELP OTHERS LEARN FROM YOU.
HONE YOUR CRAFT. RESERVE TIME FOR DELIBERATE PRACTICE. RESPECT YOUR TEAMMATES.

BE PRAGMATIC, NOT PERFECTIONIST. **DRINK**

TRY THINGS OUT. BUT DON'T FOLLOW EVERY TREND. **GOOD**

STAY DRY. **COFFEE.**

WELCOME THE ROUTINE WORK; IT'S ANOTHER OPPORTUNITY FOR AN ELEGANT SOLUTION.
REMAIN A BEGINNER. REMAIN A LEARNER.

AIM FOR SIMPLICITY.

YOU CODE FOR PEOPLE, NOT MACHINES.

1 More Learning Material

The First Aid Kit for Git with 17 videos and a handy cheat sheet!
Get it for free: www.git-tower.com/learn/git/first-aid-kit

2 10% Off for Tower

You can try Tower 30 days for free...
and save 10% with coupon code "GIT10"

Twitter @gntr
Email tg@fournova.com



TOWER
www.git-tower.com