

# Surviving with Git



How to Undo Things  
and Recover from Mistakes

Tobias Günther

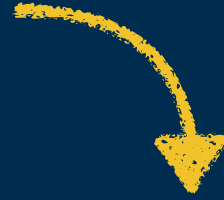
Twitter

@gntr

Email

tg@fournova.com

Making changes in a  
complex software project

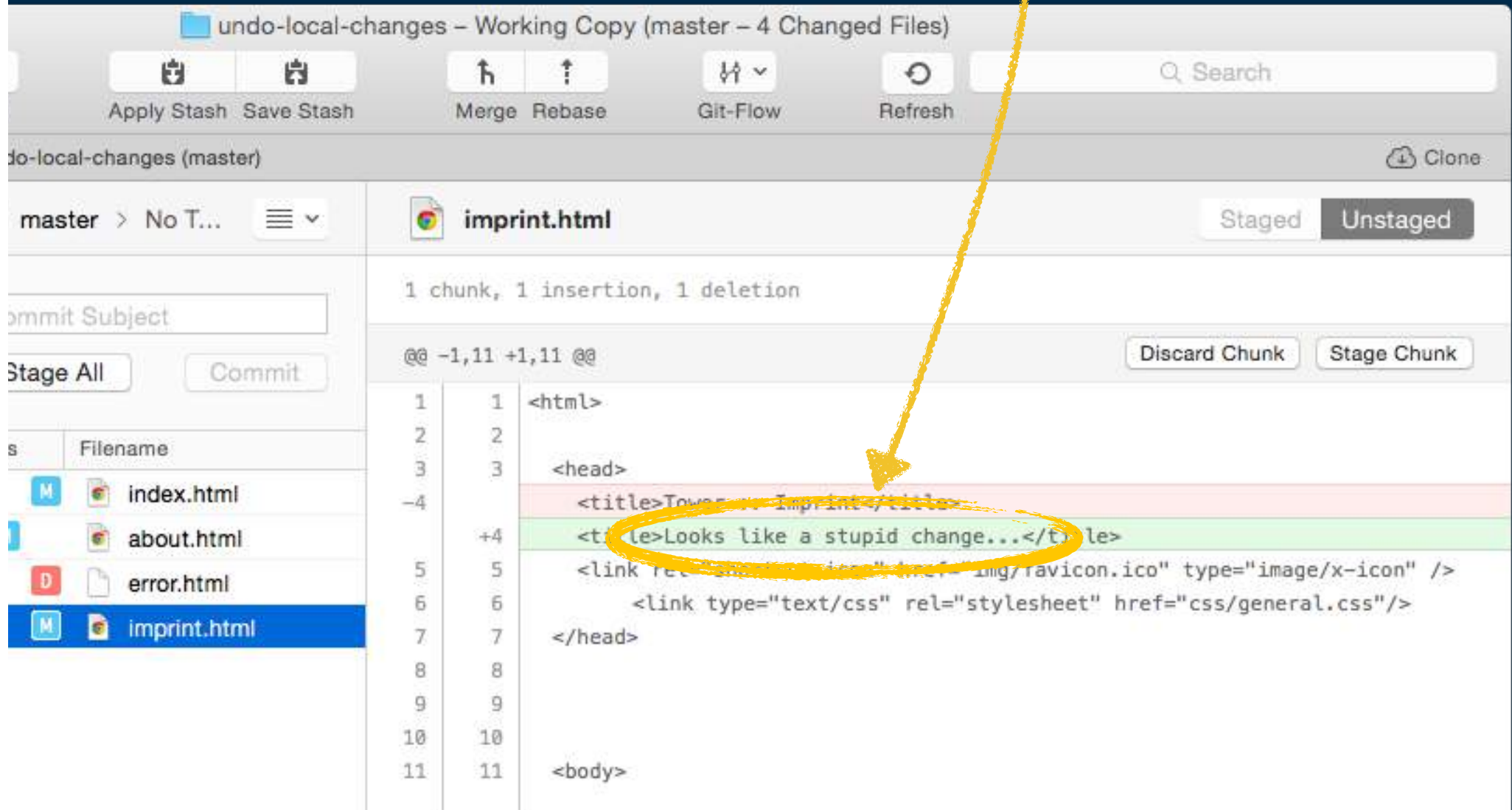


#1

# Discarding All Local Changes in a File

Apply in cases of:

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



#1

# Discarding All Local Changes in a File

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

Discarding uncommitted local changes cannot be undone!

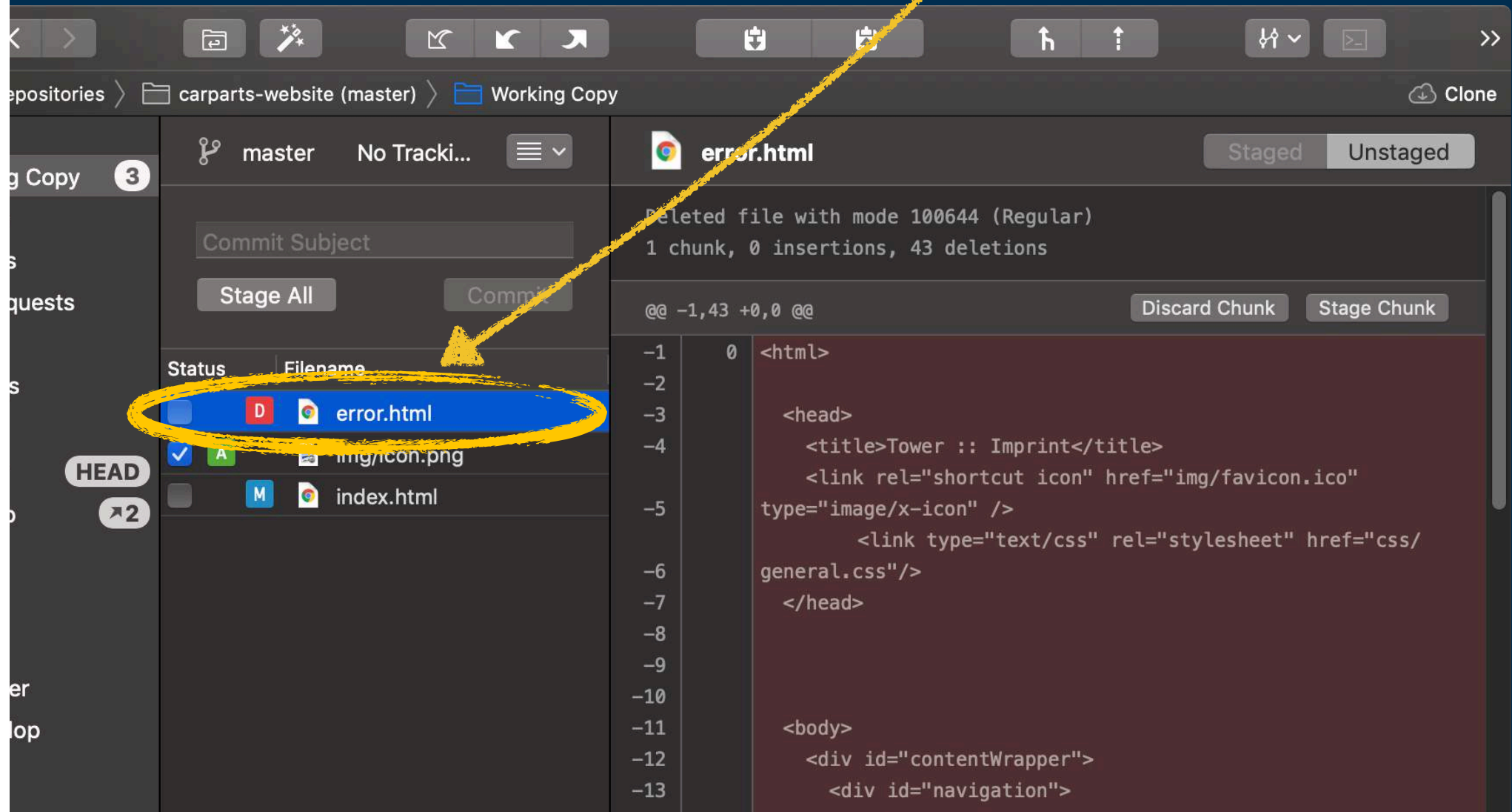


## #2

# Restoring a Deleted File

Apply in cases of:

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



#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?!!”

The screenshot shows a code editor interface with the following components:

- Top Bar:** Navigation icons (back, forward, search, etc.) and a "Clone" button.
- Breadcrumb:** Repositories > carparts-website\_orig (master) > Working Copy
- Left Sidebar:**
  - File Explorer: Shows a list of files including index.html, img/icon.png, product.html, about.html, and error.html.
  - Commit Controls: Includes a "Commit Subject" field, "Stage All", and "Commit" buttons.
  - Status: A table showing the status of files (M for modified, ? for untracked, D for deleted).
- Central Diff View:**
  - File: index.html
  - Buttons: Staged, Unstaged
  - Summary: 1 chunk, 3 insertions, 2 deletions
  - Diff Header: @@ -13,7 +13,8 @@
  - Diff Content:
    - Line 13: <div id="navigation">
    - Line 14: <ul>
    - Line 15: <li><a href="index.html">Home</a></li>
    - Line 16: -<li><a href="about.html">About</a></li> (Deletion)
    - Line 17: +<li><a href="about.html">About Us</a></li> (Insertion)
    - Line 18: +<li><a href="product.html">Product</a></li> (Insertion)
    - Line 19: <li><a href="imprint.html">Imprint</a></li>
    - Line 20: </ul>
    - Line 21: </div>
- Right Pane:**
  - Buttons: Discard Lines, Stage Lines (circled in yellow)
  - Diff Header: @@ -27,7 +28,7 @@
  - Diff Content:
    - Line 27: <a href="learn">visit our "learn"
    - Line 28: section.</a>
    - Line 29: </p>
    - Line 30: <p class="blogParagraph">
    - Line 31: -<a href="blog">Read our blog.</a> (Deletion)
    - Line 32: +<a href="blog">Read our new blog.</a> (Insertion)

## #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 left sidebar shows the file explorer with a list of files: `index.html`, `img/icon.png`, `product.html`, `about.html`, and `error.html`. The `HEAD` branch is selected. The main editor area shows the diff view for `index.html`, with a yellow circle highlighting the `HEAD` branch and the `index.html` file. The diff view shows changes to the navigation menu, with lines 16-17 being discarded and lines 18-19 being staged.

Repositories > carparts-website\_orig (master) > Working Copy

master > No Tracki... Commit Subject Stage All Commit

Status | Filename

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

index.html

1 chunk, 3 insertions, 2 deletions

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

Line	Line	Content
13	13	<div id="navigation">
14	14	<ul>
15	15	<li><a href="index.html">Home</a></li>
-16	+16	<li><a href="about.html">About</a></li>
	+17	<li><a href="about.html">About Us</a></li>
	+17	<li><a href="product.html">Product</a></li>
17	18	<li><a href="imprint.html">Imprint</a></li>
18	19	</ul>
19	20	</div>

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

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

#4

# Discarding All Local Changes

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

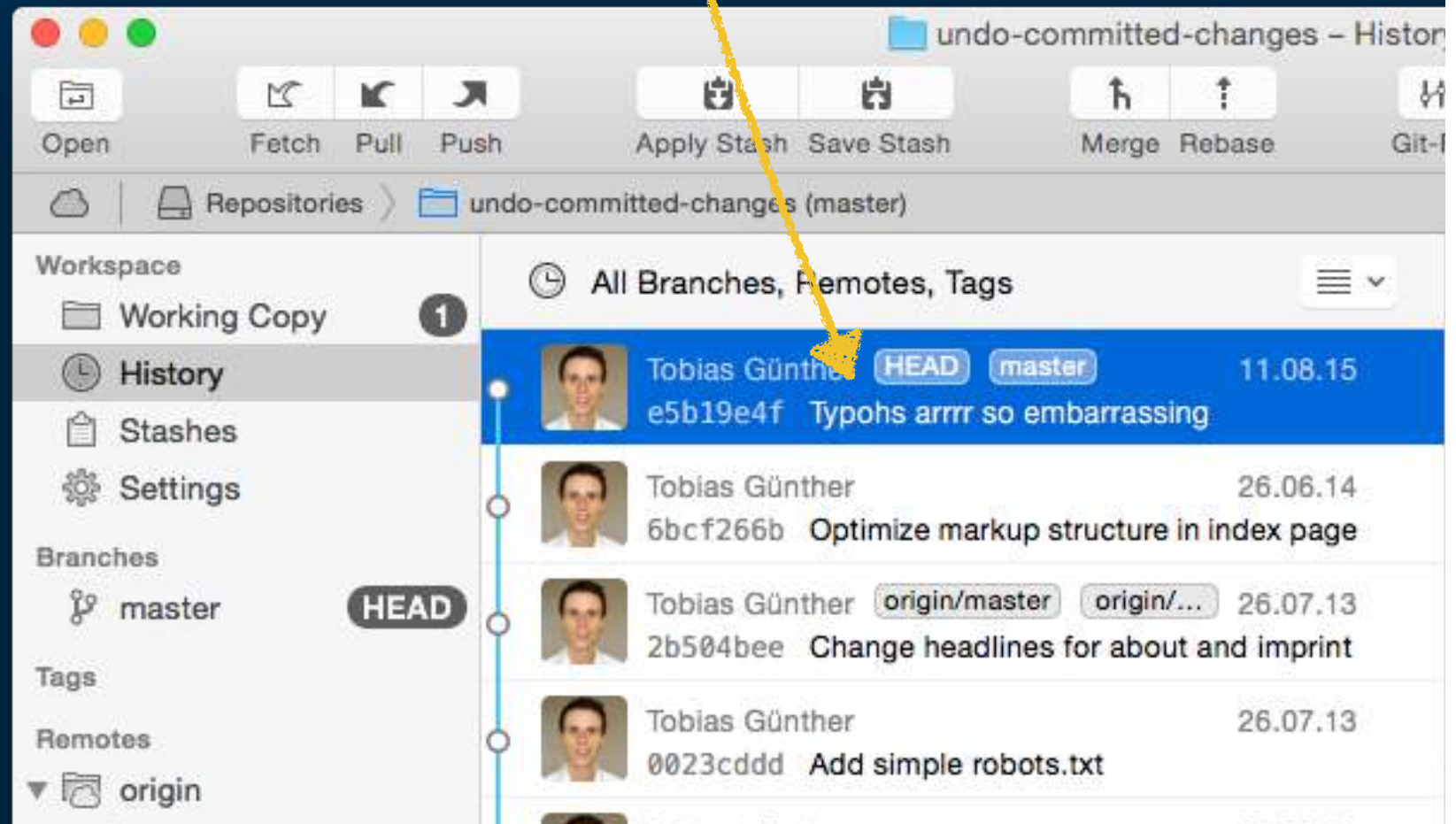
Please be careful with this command:  
discarding uncommitted local changes cannot be undone!

# #5

## Fixing the Last Commit

Apply in cases of:

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



#5

# Fixing the Last Commit

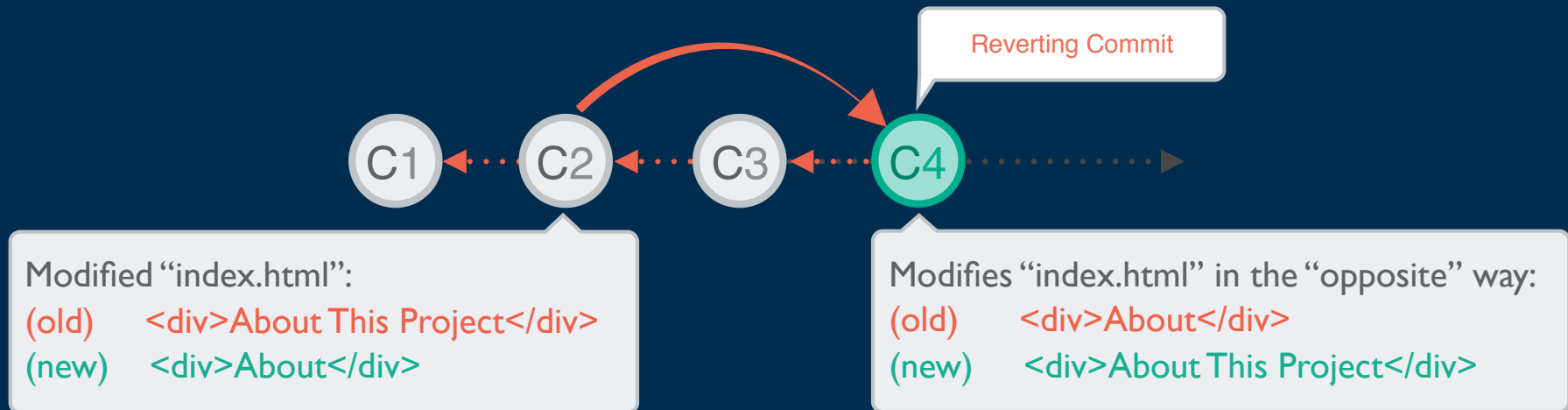
```
$ git commit --amend -m "correct"
```

“--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

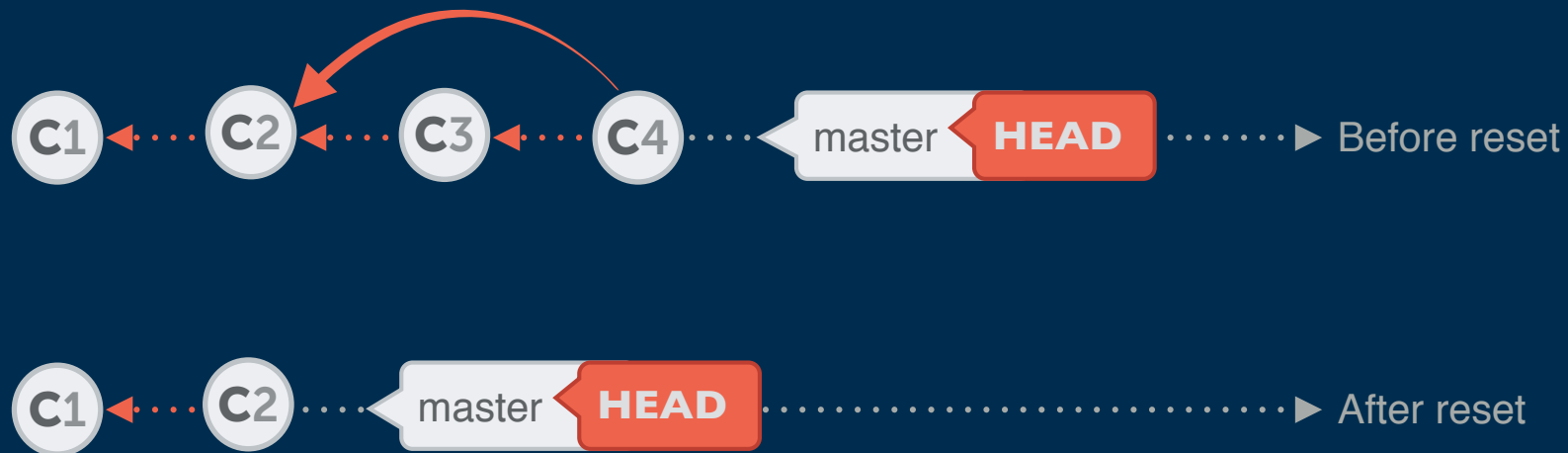
```
$ git revert 2b504bee
```

## #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

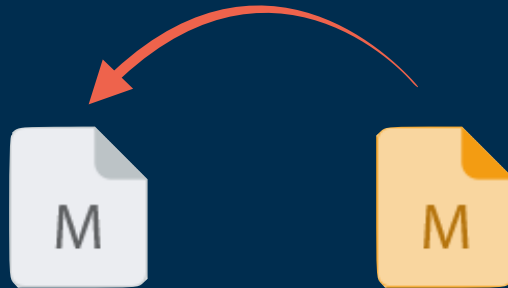
# Resetting to an Old Revision

```
$ 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

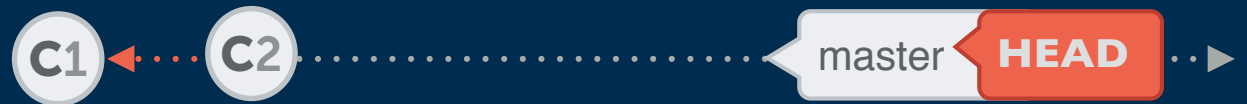
Apply in cases of:

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

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

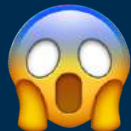


2. You reset.



3. You notice it was a bad idea.

4. You panic.



#9

# Recovering Deleted Commits

```
$ git reflog
```

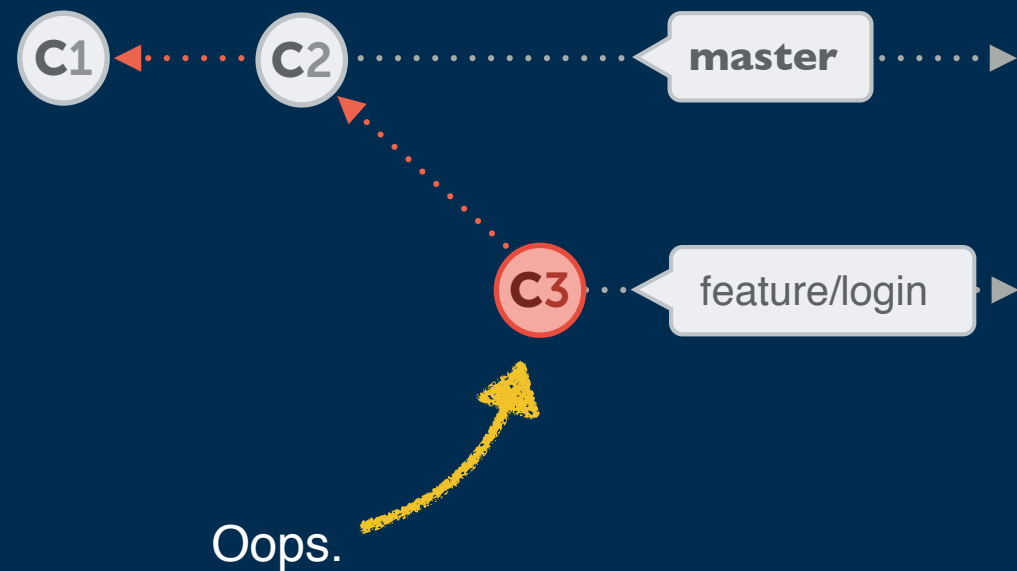
```
$ git branch happy-ending e5b19e4
```

# #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

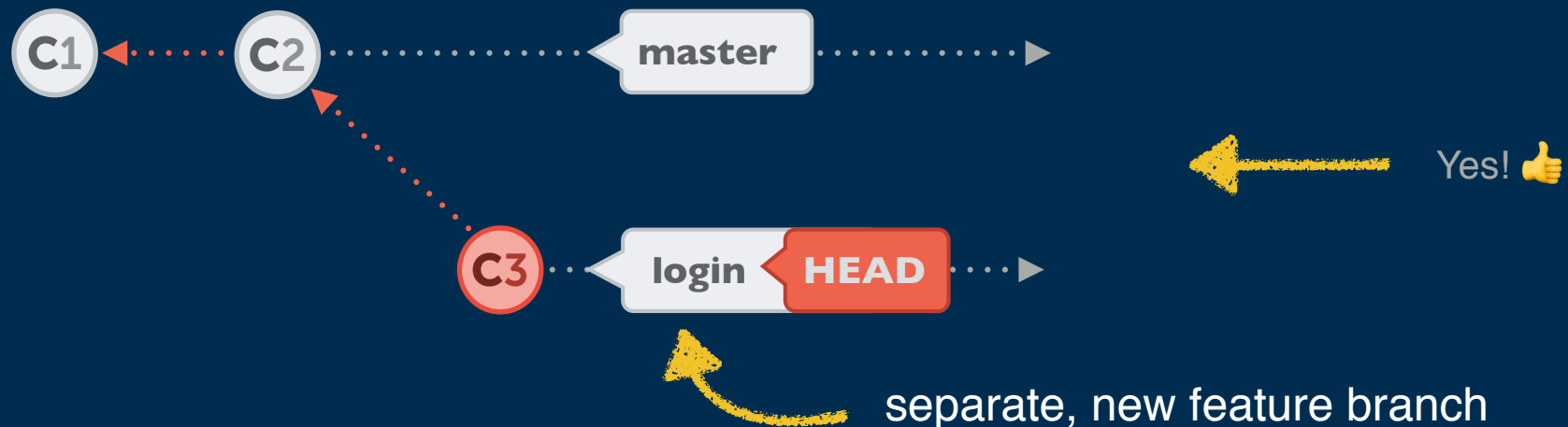
```
$ git reflog  
$ git branch develop b1c249b
```



# #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

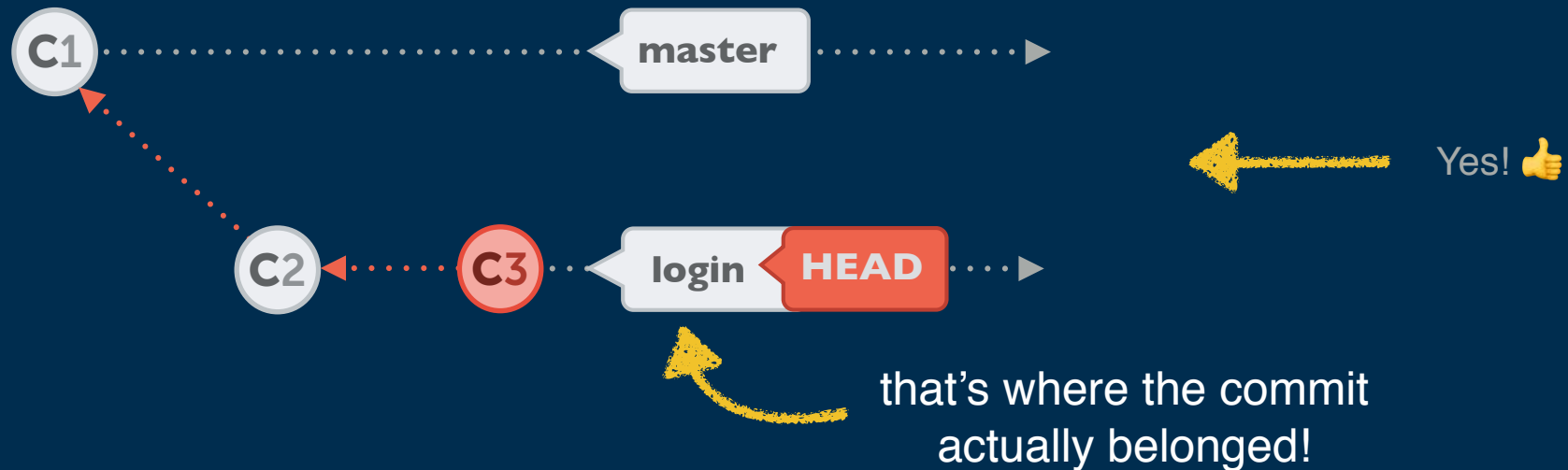
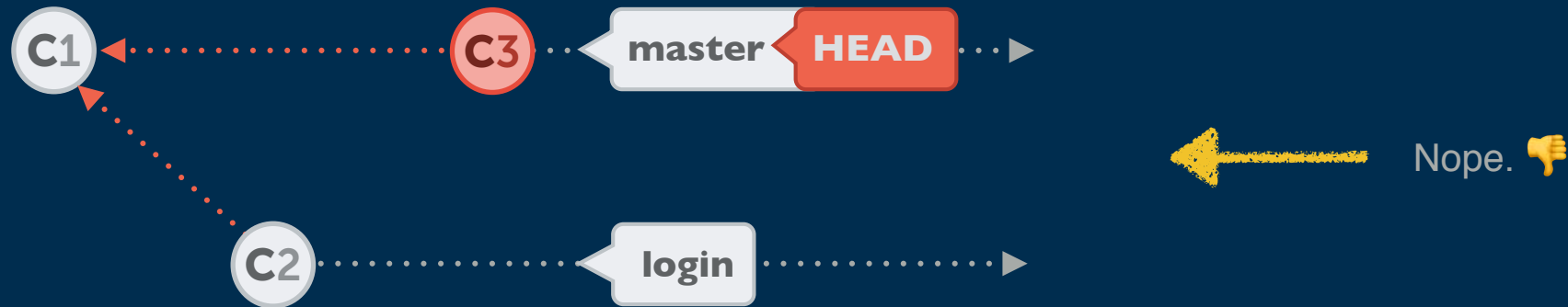
```
$ git branch feature/login  
$ git reset HEAD~1 --hard
```

## #12

# Moving a Commit to a Different Branch

Apply in cases of:

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



## #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
```

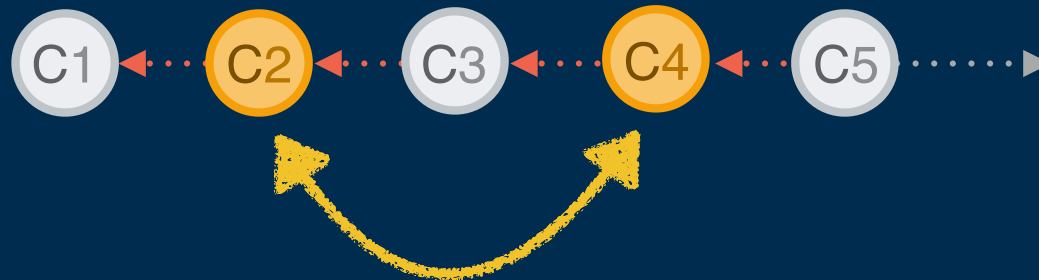
...and remove lines for unwanted commits

#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
```

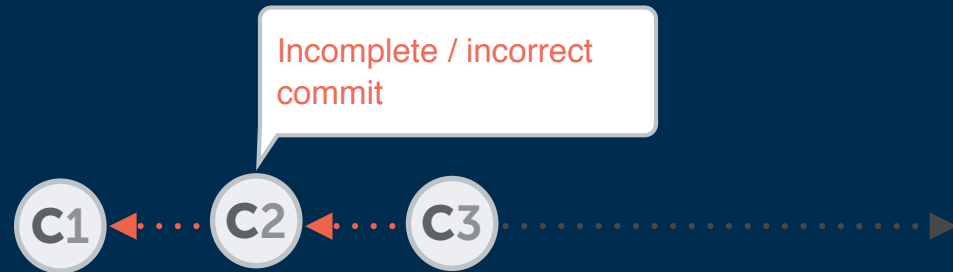
...then use “squash” option

# #16 Adding a Change to an Old Commit

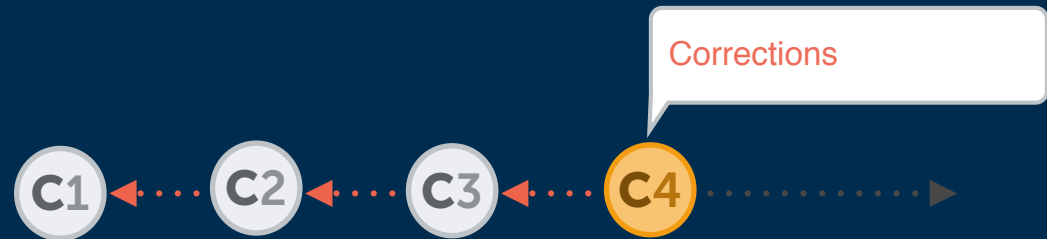
Apply in cases of:

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

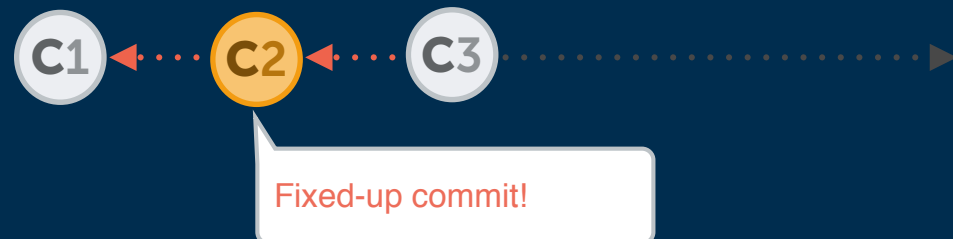
1. You forgot to commit something - or simply made a mistake!



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

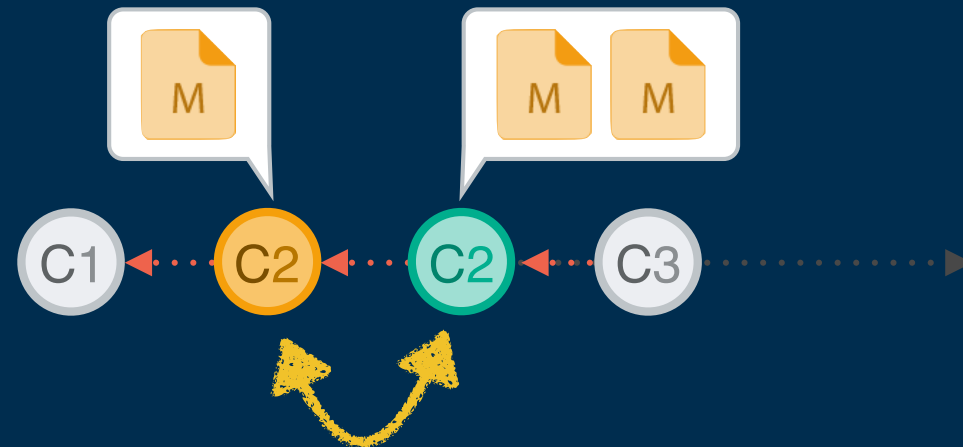
```
$ git commit --fixup 2b504bee  
$ git rebase -i --autosquash HEAD~4
```

## #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](http://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,** **DRINK**  
**NOT PERFECTIONIST.** **GOOD**  
TRY THINGS OUT. BUT DON'T  
FOLLOW EVERY TREND. **COFFEE.**  
**STAY DRY.**  
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](http://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  
Email

Tobias Günther  
@gntr  
tg@fournova.com



**TOWER**  
[www.git-tower.com](http://www.git-tower.com)