Understanding Version Control with Diffs

Tobias Günther(http://www.sitepoint.com/author/tgunther/)

May 22, 2014

Every project is made up of countless little changes. With a little luck, they will finally form a website, an app, or some other product. Your version control system keeps track of these changes. But only once you understand how to read them will you be able to track your project's progress.

Using the example of Git, the popular version control system, this article will help you understand these changes.

Diffs Represent Changes

In version control, the differences between one version and another are presented in what's called a "diff" (or a "patch"). You'll encounter diffs when working in technical surroundings like the command line, but also in graphical tools like "Tortoise Git" (http://code.google.com/p/tortoisegit/) (Win) or "Tower" (http://www.git-tower.com) (Mac) and in code hosting platforms like "GitHub" (http://www.github.com). Diffs are the most frequently-used methods of visualizing changes.

Let's learn to read diffs with an example.

Compared Files A/B

Our example diff compares two items with each other: item A and item B. In theory, A and B can be *any* file. In most cases, however, you'll be wanting to compare the same file, but in two different versions.

To set the scene, a diff output always begins by declaring which files are represented by "A" and "B".

File Metadata

Rather technical information, which you'll rarely need, follows: the file metadata.

The first two strings represent the object hashes (or "IDs") of our two files. In the background, Git not only saves versions of the project but also of each file item as an object. Hashes like the ones seen here are used to identify a file from a specific version.

Finally, the number on the right is an internal file mode identifier; '100644' is just a "normal file",'100755' specifies an executable file and '120000' represents a symbolic link.

Markers for a/b

A little later in the output, actual pieces of changes are displayed. To be able to identify which come from which version (A or B), each compared item is assigned a symbol: a minus (“-“) sign is used for version A, while a plus (“+”) sign is used for version B.

Chunk

Inspecting a 10,000 line file with only two modified lines would be a tedious process if the diff showed the entire contents of the file at once. Instead, only the portions that have actually been changed are shown. In Git, such a portion is called a “chunk” (or a “hunk”). The actual changed lines are surrounded by unchanged lines before and after the modification. These are called the “context”, because they help the user understand the context in which these modifications were made.

Chunk Header

A header prepends each of these chunks. The first part of the header informs you which lines were affected, enclosed in two “@” signs each. Let’s see which lines were modified in the first chunk of our example diff:

> From file A (represented by a “-“), 6 lines are extracted, beginning from line no. 34
> From file B (represented by a “+”), 8 lines are displayed, also starting from line no. 34

After the closing pair of “@@”, another piece of context is given: Git tries to display a method name or other contextual information about where this chunk was taken from in the file. However, this differs depending on the programming language and doesn’t work in all scenarios.

Changes

To understand a change, you must be able to see both of its states – before and after the modification. This is why Git marks the lines that were actually changed with either a “+” or a “-” sign: a line that is prepended with a “-” sign comes from A, while a line with a “+” sign comes from B.

Git kindly chooses “A” and “B” in such a way that (in most cases) you can think of A (or “-“) as “old” content and B (or “+”) as “new” content.

Our example helps us understand this:

Change #1 contains two lines prepended with a “+”. Because no counterpart in A existed for these lines (no lines with “-“), this means that the lines were added.

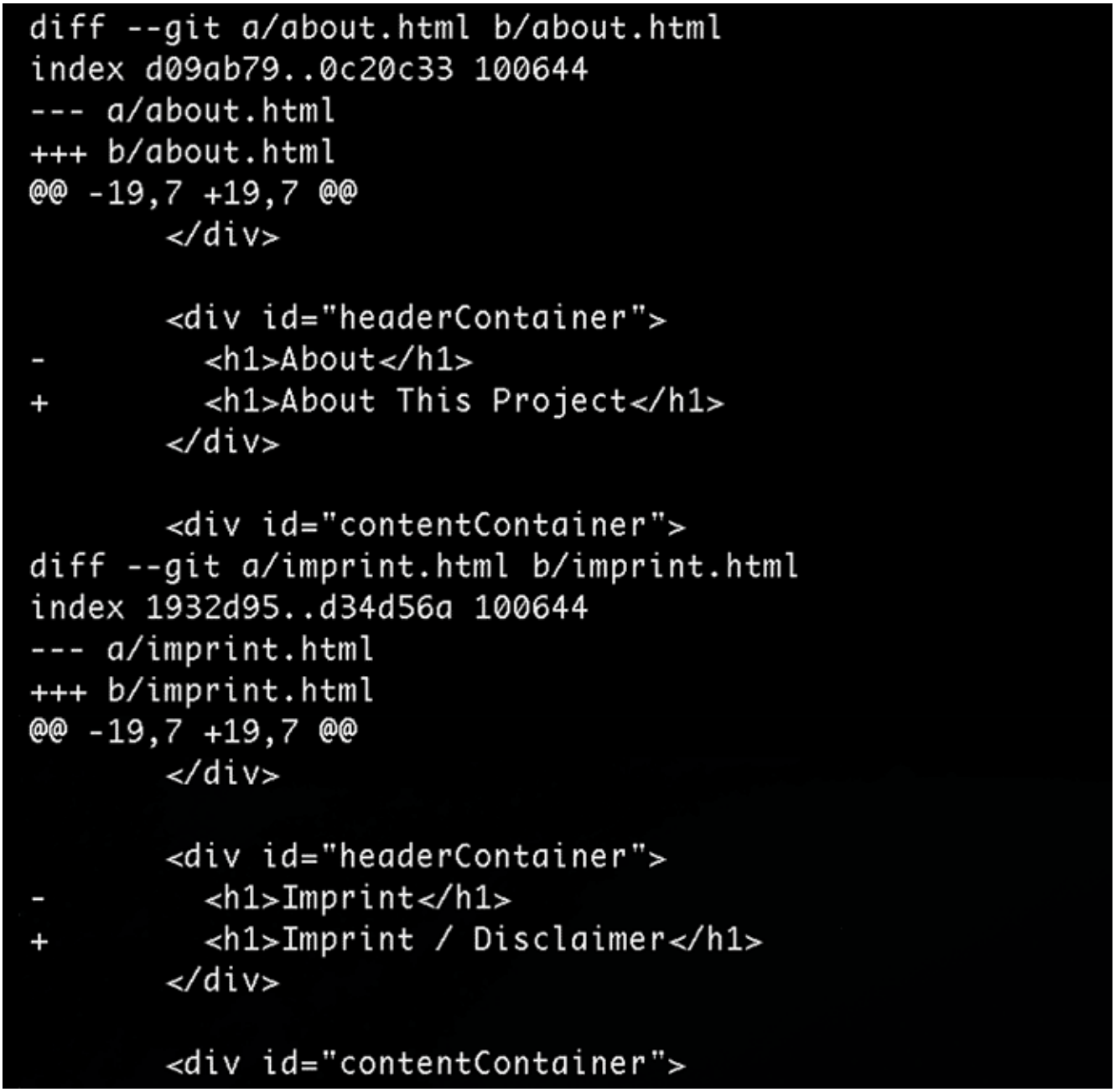
Change #2 is exactly the opposite: we have two lines marked with “-” signs in A. Since B doesn’t have an equivalent (no “+” lines), this means they were deleted.

Finally, some lines were actually modified: in Change #3, the two “-” lines were changed to look like the two “+” lines below.

You now know how to read and understand a diff output. Now let’s look at how to *generate* some of this output in Git.

Inspecting Local Changes

Before committing changes, you’ll want to see what *exactly* you did. The “git diff” command will show you all of your local changes that haven’t been added to Git’s ‘Staging Area’, yet: $ git diff



```
diff --git a/about.html b/about.html
index d09ab79..0c20c33 100644
--- a/about.html
+++ b/about.html
@@ -19,7 +19,7 @@
        </div>

        <div id="headerContainer">
-         <h1>About</h1>
+         <h1>About This Project</h1>
        </div>

        <div id="contentContainer">
diff --git a/imprint.html b/imprint.html
index 1932d95..d34d56a 100644
--- a/imprint.html
+++ b/imprint.html
@@ -19,7 +19,7 @@
        </div>

        <div id="headerContainer">
-         <h1>Imprint</h1>
+         <h1>Imprint / Disclaimer</h1>
        </div>

        <div id="contentContainer">
```

If instead you'd like to see only the changes that you've already staged, you can do so by adding the "–staged" parameter to the command.

Inspecting Committed Changes

Changes that have already been committed to the repository can be inspected with the "git log" command. By default, it only spits out an overview of the commit's metadata (like author, message, and date). If you add the "-p" flag, you can also request detailed information about the exact changes:

```
commit 2dfe283e6c81ca48d6edc1574b1f2d4d84ae7fa1
Author: Tobias Günther <tg@fournova.com>
Date:   Fri Jul 26 10:52:04 2013 +0200

    Implement the new login box

diff --git a/css/about.css b/css/about.css
index e69de29..4b5800f 100644
--- a/css/about.css
+++ b/css/about.css
@@ -0,0 +1,2 @@
+h1 {
+  line-height:30px; }
\ No newline at end of file
diff --git a/css/general.css b/css/general.css
index a3b8935..d472b7f 100644
--- a/css/general.css
+++ b/css/general.css
@@ -21,7 +21,8 @@ body {

 h1, h2, h3, h4, h5 {
   color:#ffd84c;
-  font-family: "Trebuchet MS", "Trebuchet"; }
+  font-family: "Trebuchet MS", "Trebuchet";
+  margin-bottom:0px; }

 p {
   margin-bottom:6px;}
diff --git a/error.html b/error.html
deleted file mode 100644
index 78a1c33..0000000
--- a/error.html
+++ /dev/null
@@ -1,43 +0,0 @@
-<html>
-
-  <head>
-    <title>Tower :: Imprint</title>
-    <link rel="shortcut icon" href="img/favicon.ico" type="image/x-icon" />
-               <link type="text/css" rel="stylesheet" href="css/general.css"/>
-  </head>
-
```

Comparing Branches & Revisions

Diffs are also useful when you want to understand how a certain branch (or even a specific commit) differs from another. For example, you might be interested in seeing all the changes from the “contact-form” branch that you don’t have in “master”, yet. To achieve this, use the “git diff” command as follows: $ git diff master..contact-form

But you’re not limited to comparing branches. You can even compare two arbitrary commits with each other: $ git diff 0023cdd..fcd6199

Tools for Clearer Visualization

When looking at larger and more complicated modifications, you may need all the help you can get. A dedicated diff tool application will offer an improved visualization – and will thereby make understanding changes easier.

To help you choose from the many available tools in this area, have a look at this overview of recommended apps (http://www.git-tower.com/learn/ebook/command-line/tools-services/diff-merge-tools).

Conclusion

I hope this article showed you that diffs (although they may suggest otherwise on first examination) are not just obscure ASCII art. Once you've learned how to read them, they allow you to understand how your project evolved. This makes them an essential tool for every member of a development team.

Tags: collaboration (http://www.sitepoint.com/tag/collaboration/), diffs (http://www.sitepoint.com/tag/diffs/), git (http://www.sitepoint.com/tag/git/), software development (http://www.sitepoint.com/tag/software-development-3/), version control (http://www.sitepoint.com/tag/version-control-2/), versioning (http://www.sitepoint.com/tag/versioning/), web development (http://www.sitepoint.com/tag/web-development/)

Tobias Günther (http://www.sitepoint.com/author/tgunther/)

(https://twitter.com/@gntr) (https://plus.google.com/u/0/101562701283967902986)

Tobias Günther is the author of the free online book "Learn Version Control with Git - A step-by-step guide for the complete beginner" (http://www.git-tower.com/learn). He's also part of the team behind "Tower" (http://www.git-tower.com), the popular Git client for Mac.

Faster Websites with Nginx

Chris Lea

★★★★½

(https://www.sitepoint.com/premium/cou
websites-with-nginx-2757/?
utm_source=sitepoint&utm_medium=relat

1:17:14

Build a Corporate Website with Joomla

Kray Mitchell

★★★★☆

(https://www.sitepoint.com/premium/cou
a-corporate-website-with-joomla-
2726/?
utm_source=sitepoint&utm_medium=relat

3:07:36

JavaScript: Next Steps

M. David Green

★★★★★

(https://www.sitepoint.com/premium/cou
next-steps-2921/?
utm_source=sitepoint&utm_medium=relat

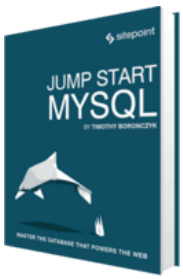
BOOKS >
(https://www.sitepoint.com/prem
q=&content_types[]=Book&utm_s

Level Up Your Web Apps With Go

Mal Curtis

★★★★½

(https://www.sitepoint.com/premium/boo
up-your-web-apps-with-go/?
utm_source=sitepoint&utm_medium=relat



Jump Start MySQL

Timothy Boronczyk

★★★★½

(https://www.sitepoint.com/premium/boo
start-mysql/?
utm_source=sitepoint&utm_medium=relat

Deliver First-Class Websites: 101 Essential Checklists

Shirley Kaiser

★★★★☆

(https://www.sitepoint.com/premium/boo
first-class-websites-101-essential-
checklists/?

utm_source=sitepoint&utm_medium=relat

SCREENCASTS > (https://www.sitepoint.com/prem
q=&content_types[]=ScreenCast&

Working with a Database on the Command Line

Dr. Richard Stibbard

(https://www.sitepoint.com/premium/tutc
with-a-database-on-the-command-line/?
utm_source=sitepoint&utm_medium=relat

Creating MySQL Databases and Tables From the CLI

Dr. Richard Stibbard

(https://www.sitepoint.com/premium/tutc
mysql-databases-and-tables-from-the-cli/?
utm_source=sitepoint&utm_medium=relat

MySQL on the Command Line

Dr. Richard Stibbard

(https://www.sitepoint.com/premium/tutc
on-the-command-line/?
utm_source=sitepoint&utm_medium=relat

Contribute (/write-for-us/)

Visit

SitePoint Home (/)
Forums (https://www.sitepoint.com/community/)
Newsletters (/newsletter/)
Premium (/premium/)
References (/sass-reference/)
Shop (https://shop.sitepoint.com)
Versioning (https://www.sitepoint.com/versioning/)

Connect

(http://www.sitepoint.com/feed/) (/newsletter/) (https://www.facebook.com/sitepoint) (http://twitter.com/sitepointdotcom) (https://plus.google.com/+sitepoint)