INTRODUCTION TO THE COMMAND LINE

Excerpt from:
Learning Web Design, 5e

by Jennifer Robbins

Author's Note: *This article first appeared in Learning Web Design, 5e, released in 2018. It was removed from the 6th edition, but I have made it available here as supplemental material. Keep in mind that some details may be out of date.*

IN THIS ARTICLE

You probably use a computer with a graphical user Interface (GUI), with icons that stand for files and directories, pull-down menus full of options, and intuitive actions like dragging files from folder to folder or into the trash.

Computer users in the '60s and '70s didn't have that luxury. The only way to get a computer to perform a task was to type in a command. Despite our fancy GUIs, typing commands into a command-line terminal is far from obsolete. In fact, the more experienced you become at web development, the more likely it is you'll dip into the command line for certain tasks. If you are already a programmer, the command line will be nothing new.

The command line is still popular for a number of reasons. First, it is useful for accessing remote computers, and developers often need to access and manage files on remote web servers. In addition, it is easier to write a program for the command line than a standalone application with a GUI, so many of the best tools for optimizing our workflow exist as command-line programs only. A lot of those tools can be used together in a pipeline for accomplishing complex tasks.

The time- and sanity-saving benefits are powerful incentives to take on the command line. Trust me: if you can learn all those elements and style properties, you can get used to typing a few commands.

THE COMMAND-LINE TERMINAL

Every Mac and Linux machine comes installed with the Terminal application for using the command line. On macOS, you will find the Terminal program in Applications → Utilities (FIGURE A).

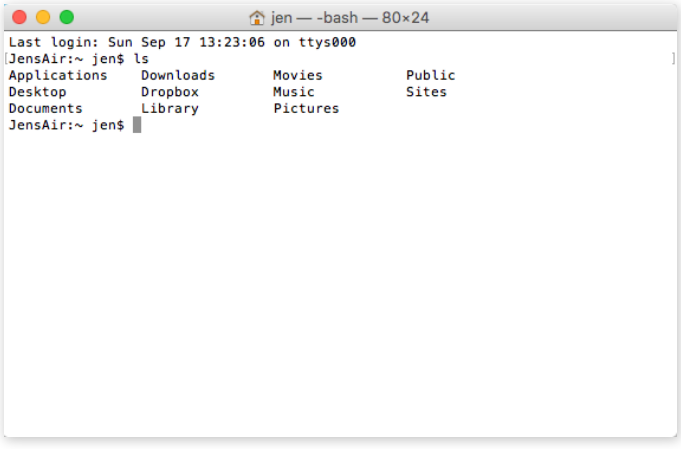

FIGURE A. The Terminal window in macOS.

The program that interprets the commands you type is called a shell (visual interfaces are also technically a shell; they're just fancier). Linux machines use a shell called bash. The macOS terminal uses a shell called Zsh (Z shell) (see Note) which is similar to bash but offers more expandability and customization.

NOTE

The macOS terminal used bash until it switched to Zsh in 2019 with the release of the Catalina OS.

The default command-line tool on Windows is Windows Terminal (most easily accessed with Search). It can run multiple shells, including PowerShell and Git Bash (a bash-like shell that makes it easier to interface with Git. Another option is to install a bash-based shell emulator like Cygwin (*cygwin.com*) or cmder (*cmder.net*).

GETTING STARTED WITH COMMANDS

[***Author's note:*** *Examples in this section were written when macOS used the bash shell. Prompts may look different in the current version of the Terminal app, but the underlying lesson is the same.]*

When you launch a Terminal window, the first thing you see is a command-line prompt, which is a string of characters that indicates the computer is ready to receive your command:

```
$: _
```

The dollar sign ($) is common, but you may see another symbol in your terminal program (see Tip). The underscore in this example stands for the cursor position, which may appear as a little rectangle or a flashing line.

The complete prompt that I see in Terminal begins with my computer's name ("JensAir") and an indication of the working directory—that is, the directory the shell is currently looking at. In GUI terms, the working directory is the folder you are "in." In this example, the tilde (**~**) indicates that I am looking at my root User directory. The "jen" before the prompt character is my username. In future examples, I will abbreviate the prompt to simply **$:**.

```
JensAir:~ jen$: _
```

When you see the prompt, type in a command, and hit Enter. The computer executes the command and gives you a new prompt when it is finished. It is very no-nonsense about it. For some commands, there may be feedback or information displayed before the next prompt appears. Sometimes everything happens behind the scenes, and all you see is a fresh prompt.

When you're learning about the command line, it is common to start with the built-in commands for navigating the file system, tasks typically handled by the Finder on the Mac and My Computer on Windows. Because they are fairly intuitive, that's where I'm going to start my simple command-line lesson as well.

A nice little utility to try as a beginner is **pwd** (for "print working directory"), which displays the complete path of the working (current) directory. To use it, simply type **pwd** after the prompt. It's a good one to try first because you can't break anything with it, but for seasoned users, it is useful for figuring out exactly where you've landed if you're disoriented. The forward slash indicates that this path starts at the root directory for the entire computer.

```
$: pwd
/Users/jen
```

Here's another easy (and low-risk!) example. Typing the **ls** command at the prompt returns a list of the files and directories in the working directory (*/Users/jen*). You can compare it to the Finder view of the same folder in FIGURE B. They are two ways of looking at the same thing, just as *directory* and *folder* are two terms for the same thing depending on your view.

```
JensAir:~ jen$ ls
Applications    Downloads    Movies    Public
Desktop         Dropbox      Music     Sites
Documents       Library      Pictures
JensAir:~ jen$
```

TIP

You can customize the appearance of Terminal by selecting **Preferences → Profile** and changing the settings. If you want to keep yourself amused, you can change the prompt character from $ to the character of your choice, including an emoji (*osxdaily.com/2013/04/08/add-emoji-command-line-bash-prompt/*).

NOTE

Your user directory is the default root directory in Terminal and is represented by a tilde (~) in the prompt, as we saw in the previous example.

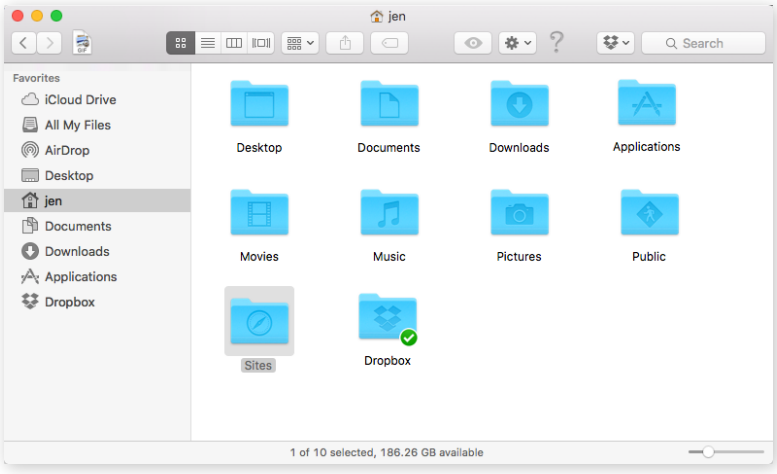

FIGURE B. Finder view of the *jen* home folder.

Dotfiles

There are some files on your computer that are kept hidden in the Finder view. These files, known as dotfiles, start with (you guessed it) a dot, and they tend to handle information that is intended to function behind the scenes. If you type **`ls -a`** (**`-a`** is shorthand for "all"), you can reveal the dotfiles lurking in a directory. In macOS, it is possible to configure Finder to show dotfiles as well, but for most users' purposes, hidden is a good thing.

Some utilities, like **`pwd`**, require only their name at the prompt to run, but it is more common that you'll need to provide additional information in the form of flags and arguments. A flag changes how the utility operates, like an option or a preference. It follows the command name and is indicated by a single or double dash (**`-`**). In many cases, flags can be abbreviated to just their first letter because they are used in context with a particular utility. For example, I can modify the **`ls`** utility with the **`-l`** flag, which instructs the computer to display my directory contents in "long" format, including permission settings and creation dates:

```
JensAir:~ jen$ ls -l
total 0
drwxr-xr-x     5 jen  staff   170 Jul  8  2016 Applications
drwx------    57 jen  staff  1938 Sep 11 09:47 Desktop
drwx------    26 jen  staff   884 May 18 11:34 Documents
drwx------+ 151 jen  staff  5134 Sep  3 15:47 Downloads
drwx------@  48 jen  staff  1632 Aug 16 16:34 Dropbox
drwx------@  72 jen  staff  2448 Jul 15 11:21 Library
drwx------    22 jen  staff   748 Oct  6  2016 Movies
drwx------    12 jen  staff   408 Sep 29  2016 Music
drwx------    14 jen  staff   476 Oct 13  2016 Pictures
drwxr-xr-x     6 jen  staff   204 May  6  2015 Public
drwxr-xr-x    11 jen  staff   374 Jul 10  2016 Sites
JensAir:~ jen$
```

An argument provides the specific information required for a particular function. For example, if I want to change to another directory, I type **cd** (for "change directory") as well as the name of the directory I want to go to (see **Tip**). To make my Dropbox directory the new working directory, I type this:

```
JensAir:~ jen$: cd Dropbox
```

After I hit Enter, my prompt changes to **JensAir:Dropbox jen$**, indicating that I am now in the Dropbox directory. If I entered **ls** now, I'd get a list of the files and folders contained in the Dropbox folder.

To go up a level, and get back to my home user directory (**~**), I can use the Unix shorthand for "go up a level": **..** (remember that from your URL path lesson?). The returned prompt shows I'm back at my root directory (**~**).

```
JensAir:Dropbox jen$ cd ..
JensAir:~ jen$
```

Some other useful file-manipulation commands include **mv** (moves files and folders), **cp** (copies files), and **mkdir** (creates a new empty directory). The **rm** command removes a file or folder in the working directory. Be careful with this command, however, because it doesn't just move files to the Trash; it removes them from your computer entirely (see the **"A Word of Caution"** note).

Another handy command is **man** (short for *manual*), which displays documentation for any command you pass to it. For example, **man ls** shows a description of the **ls** (list) command and all of its available flags. Some man pages are long. To move down in the scroll, hitting the Return key moves you down one line at a time. To move down a page at a time, hit fn+down arrow on a Mac or Shift+Page Down on Linux. To go back up a page, it's fn+up arrow or Shift+Page Up, respectively. Finally, to quit out of the man page, type **q** to return to the prompt.

LEARNING MORE

Not surprisingly, these commands are just the tip of the tip of the iceberg when it comes to command-line utilities. For a complete list of commands that can be used with bash and Zsh, see "An A–Z Index of the Bash Command Line for Linux" at *ss64.com/bash/*. You'll pick these up on an as-needed basis, so don't get overwhelmed. In addition, as you start installing and using new tools like the ones listed in this chapter, you'll gradually learn the commands, flags, and arguments for those too. All part of a day's work!

Clearly, I don't have the space (and if I'm being honest, the experience) to write a comprehensive tutorial on the command line in this chapter, but you will find books and plenty of tutorials online that can teach you. I found Michael Hartl's tutorial "Learn Enough Command Line to Be Dangerous" to be thorough and accessible if you are starting from square one (*www.learnenough.com/command-line-tutorial#sec-basics*). I also recommend the series of tutorials from Envato Tuts+, "The Command Line for Web Design" (*webdesign.tutsplus.com/series/the-command-line-for-web-design--cms-777*).

TIP

On the macOS, Terminal is well integrated with the Finder. If you need to enter a pathname to a directory or a file, you can drag the icon for that file or folder from the Finder to the Terminal, and it will fill in the pathname for you.

TIP

Typing **cd** followed by a space always takes you back to your home directory.

A WORD OF CAUTION

The command line allows you to muck around in critical parts of your computer that your GUI graciously protects from you. It's best not to type in a command if you don't know exactly what it does and how it works. Make a complete backup of your computer before you start playing around with command line so you have the peace of mind that your files are still available if something goes horribly wrong.