Web Development 4: Week One

Git

Using Terminal

- Download iTerm
- http://www.iterm2.com/
- Move it to your Applications folder

Oh My Zsh

- http://ohmyz.sh/
- http://en.wikipedia.org/wiki/Z_shell
- Zsh (zshell) - command line interface (cli) language
- CLI input: curl -L http://install.ohmyz.sh | sh

CLI commands

- ls - list
- la - list all
- vi - open editor
    - i - edit file
        - CTRL + c - exit
            - :q - quit
            - :wq - save and quit
- cd - change directory
- mkdir - make directory
- touch - create file
- zsh - execute file

Configure Zsh

- https://github.com/robbyrussell/oh-my-zsh

- Choose Agnoster theme

Color Themes

- Solarized Dark Color Scheme
  - http://ethanschoonover.com/solarized
- git clone git://github.com/altercation/solarized.git
  - navigate through to item readme.md

Powerline Symbols

- echo "<2b80> ± <2b60> ➦ ✔ ✘ <26a1>”
- Install Menlo for Powerline
    - https://gist.github.com/qrush/1595572
- Install Source Code Pro for Powerline
    - https://github.com/Lokaltog/powerline-fonts/tree/master/SourceCodePro

Remove username

- export DEFAULT_USER=robertriggs

Non-ASCII Font

- Terminal.. Preferences.. Text

- Set Non-ASCII font to Sauce Code powerline

Install Powerline Theme

- https://github.com/jeremyFreeAgent/oh-my-zsh-powerline-theme

- zsh the install file within the theme

Git

- Git: http://git-scm.com/
- Free open source version control system
- Fast
- Works well with Github: https://github.com/

Setting up Git

- https://help.github.com/articles/set-up-git

- config user name and email

GitHub

- Sign in and create new repo on github.com
- webdev4
- Web Dev 4, Summer 2014
- do NOT initialize repo

Getting Started with Git

- http://git-scm.com/book/en/Getting-Started
- Download Git: http://git-scm.com/download/mac

- If won't install, try here:
  - http://git-scm.com/book/en/Getting-Started-Installing-Git

Local Repo

- Set up folder

- Git clone github url

github

- Create a new repository on the command line
- touch README.md
- git init
- git add README.md
- git commit -m "first commit"
- git remote add origin https://github.com/robr24/webdev4.git
- git push -u origin master
- Push an existing repository from the command line

- git remote add origin https://github.com/robr24/webdev4.git
- git push -u origin master

How Git Works

Git Has Integrity

Everything in Git is check-summed before it is stored and is then referred to by that checksum. This means it's impossible to change the contents of any file or directory without Git knowing about it. This functionality is built into Git at the lowest levels and is integral to its philosophy. You can't lose information in transit or get file corruption without Git being able to detect it.

The mechanism that Git uses for this checksumming is called a SHA-1 hash. This is a 40-character string composed of hexadecimal characters (0–9 and a–f) and calculated based on the contents of a file or directory structure in Git. A SHA-1 hash looks something like this:

```
24b9da6552252987aa493b52f8696cd6d3b00373
```

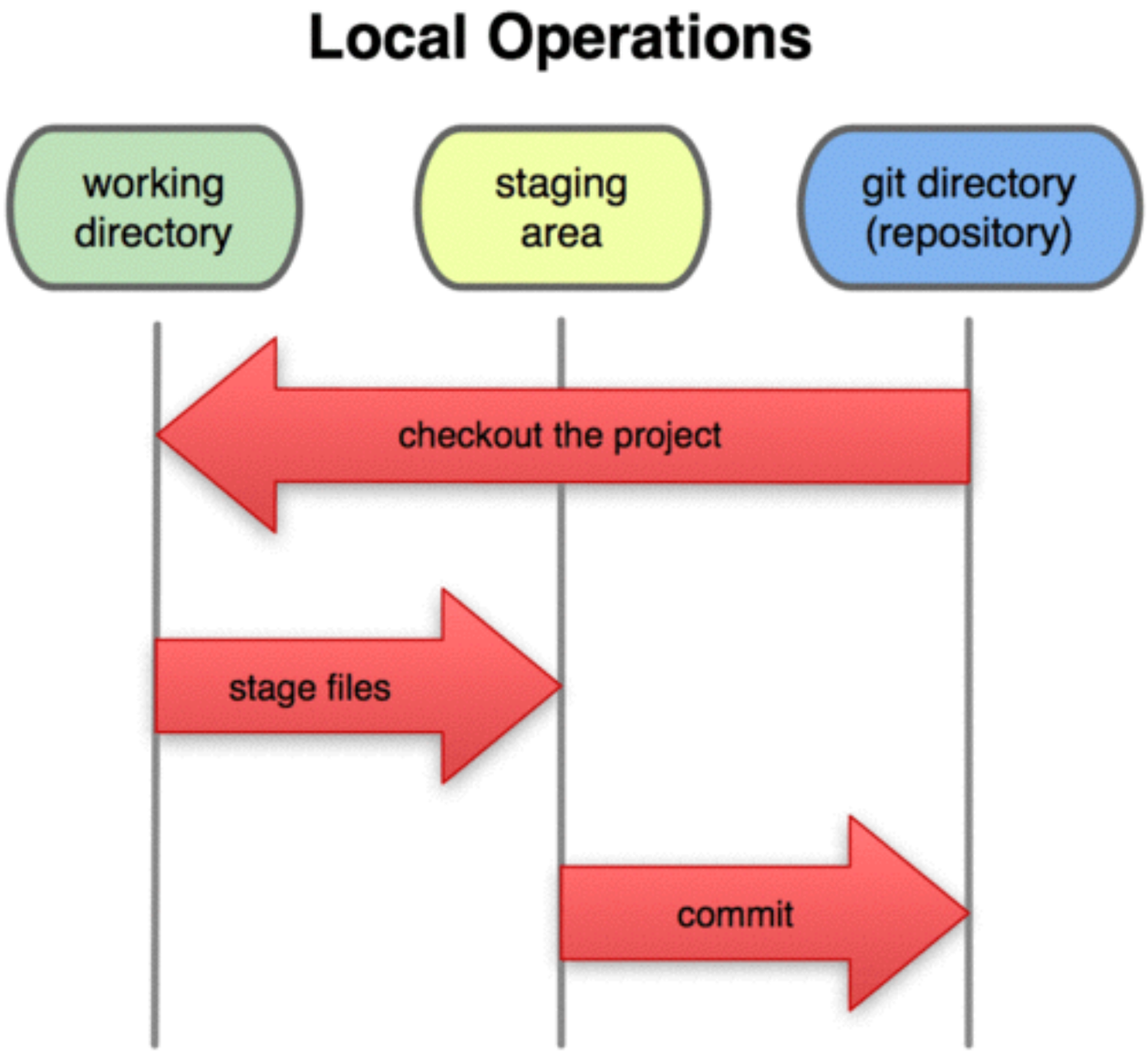
You will see these hash values all over the place in Git because it uses them so much. In fact, Git stores everything not by file name but in the Git database addressable by the hash value of its contents.

The Three States

The Three States

Now, pay attention. This is the main thing to remember about Git if you want the rest of your learning process to go smoothly. Git has three main states that your files can reside in: committed, modified, and staged. Committed means that the data is safely stored in your local database. Modified means that you have changed the file but have not committed it to your database yet. Staged means that you have marked a modified file in its current version to go into your next commit snapshot.

This leads us to the three main sections of a Git project: the Git directory, the working directory, and the staging area



Basic Git Workflow

The basic Git workflow goes something like this:

. You modify files in your working directory.
. You stage the files, adding snapshots of them to your staging area.
. You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.