COSC 419: Topics in Computer Science

Fall 2018 – Introduction to Full-Stack Development

Instructor: Matthew Fritter

matt@frit.me

Course Topics

- In this course, students will attain an understanding of the following components of the Web Server Software Stack:
- This includes Front-End development using HTML5, CSS, JavaScript, and common libraries such as JQuery and Bootstrap
- Back-End technologies, including the Apache Webserver, Laravel PHP Framework, and the WordPress Content Management System
- Use and integration of common database technologies, such as MySQL and SQLite, for developing database-driven web applications
- Common security exploits and ways of mitigating them, both at the server and client side
- Automation of server administration tasks, backups, and recovery using Chron and Bash

Evaluation

Item	Percentage of Final Grade
Quizzes x4	20% (5% each)
Labs	50%
Final Exam	30%

- Quizzes will be short, in-class tests to gauge your understanding of the course material. ***They will be announced the week prior to the quiz*** to give you time to catch up on notes and lab material
- If you have been following along with the notes and labs, you should be able to do well
- If a quiz is missed for a valid reason (family emergency, doctor's note), the weight for the missed quiz will be shifted to the final exam. Otherwise, a grade of zero will be given
- Unless otherwise noted in the course schedule, ***labs are due one week from the lab date, prior to the start of the next lab*** (6:30PM Wednesday)
- Labs will be accepted up to 24 hours past the due date with a 10% deduction in mark. Past 24 hours, a grade of zero will be given without valid proof of hardship

Labs

- For this course, each student will receive the login credentials for a remote server with a base CentOS 7 install. You will have complete, root access to the entirety of your server
- Over the course of the term, students will install and configure a web software stack on their server, build web applications using a framework, harden their server security, and automate their server management
- Should your server become unavailable for any reason (downtime, misconfiguration, etc), please contact me and I will bring it back online
- Marking for labs will be based on both outward performance (what the user would/should see when they visit your site), as well as the background and stack (the code I see when I log into your server)

Plagiarism and Academic Dishonesty

- There are a myriad of resources available for web developers on the web – I encourage you to make use of them
- Reuse of ***short*** snippets of code from sources such as Stack Exchange or the MDN is acceptable if attribution is given. A comment with a link to the original source will suffice
- However, reuse of large sections of code or script will be considered plagiarism. If you have any concerns that you may be plagiarizing, please talk to me or email me ***before*** you submit your work. Any instances of plagiarism from other students or web resources in submitted work will be forwarded to the department head
- To be very clear here: ***It is easier to ask permission than beg forgiveness! (With apologies to Grace Hopper)*** Once an academic honesty report moves up the chain to administration, there is little to nothing I can do to help you. ***Don't put yourself in that situation!***

Required Texts and Errata

- There are no required texts for this course
- It is recommended that students bring a laptop or tablet to class, or use a lab machine to facilitate taking notes and participating in class demonstrations
- Slides and assignments will be posted to the website listed on the course outline

Questions about the Outline?

★ So What is Full-Stack Development, Anyways?

- Full-Stack development has a number of definitions, but for the purposes of this course, Full-Stack development is defined as:

  A development technique in which an individual developer works on all components of the Software Stack, including the front end, back end, database, and underlying server system

- Full-Stack development has become particularly popular in the web industry – Full-Stack developers are in high demand (currently 8 job postings on Accelerate Okanagan alone, another 1,350 across Canada)
- A large portion of tech employees at major firms like Google and Facebook are effectively Full-Stack Developers
- Often associated with Agile/Scrum, rapid prototyping, and other flexible/fast software development frameworks
- In short, ***the Full-Stack Developer is a generalist***

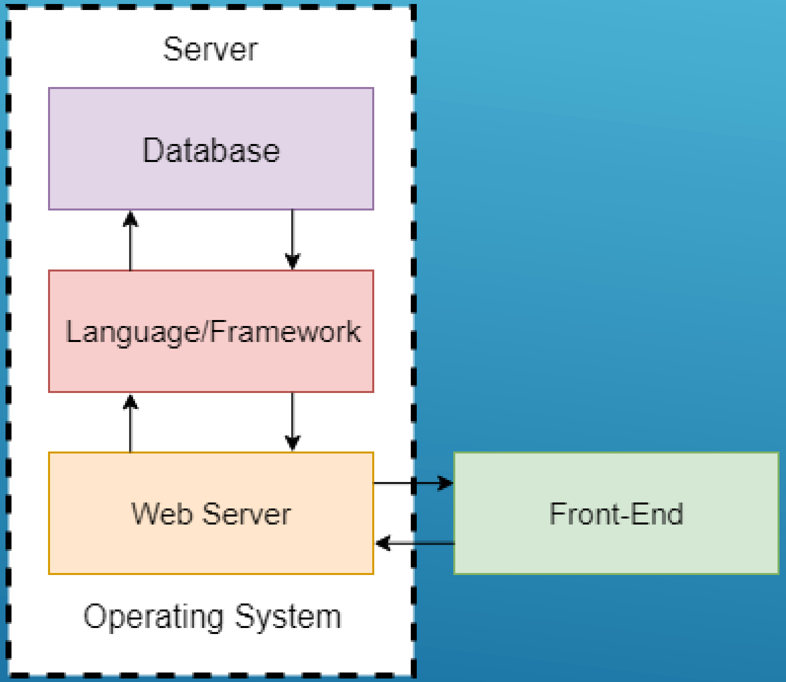
★ Why is Full-Stack Development Powerful?

- ***You are in total control of your software ecosystem***: Being adept at all layers of the software stack, you can pick and choose your software to match your requirements and your skills
- ***You are flexible***
  - Client wants you to add a new feature by next week's launch
    - *You can write backend and frontend code, you can implement it*
  - Your database gets wiped out, and you need it back online ASAP
    - *You wrote a backup script and understand how the database works because you modeled it, you can recover it*
  - A security flaw has been found in your OS, and you need to update
    - *You know how to maintain your system, you can patch it*
  - You need to change your web domain for rebranding reasons
    - *You set up the web server, you can configure it*

Full-Stack Developers vs. Specialists

- A Full-Stack Developer does not necessarily replace field specialists, but can work alongside them
- The reality is that ***there aren't enough hours in the day to master every aspect of Full-Stack Development to the degree of a dedicated specialist***. On the bleeding edge of technological development, the domain knowledge that specialists like database engineers, UX designers, and networking engineers provide is critical
- In these kind of environments, the Full-Stack Developer can help unify the work done by specialists into a cohesive hole, and fill gaps between knowledge domains

★ The Core Components of the Stack



1. Server Operating System
2. Web Server Software
3. Front-end
4. Back-end Language/Framework
5. Database

The Operating System

- ***The OS is the basis for the software stack***, and the choice of OS will influence many of your other software choices
- Web server operating systems have different feature sets than typical consumer operating systems, such as support for large number of user connections, higher maximum memory sizes, and greater fault tolerance
- Depending on the survey, Unix-based server operating systems account for between 96% and 67% of the market (largely Linux systems), with Microsoft Windows Server products making up the remainder
- Many server-oriented operating systems come with guarantees of long-term support. For example, Ubuntu 18.04 LTS is to be supported until 2023

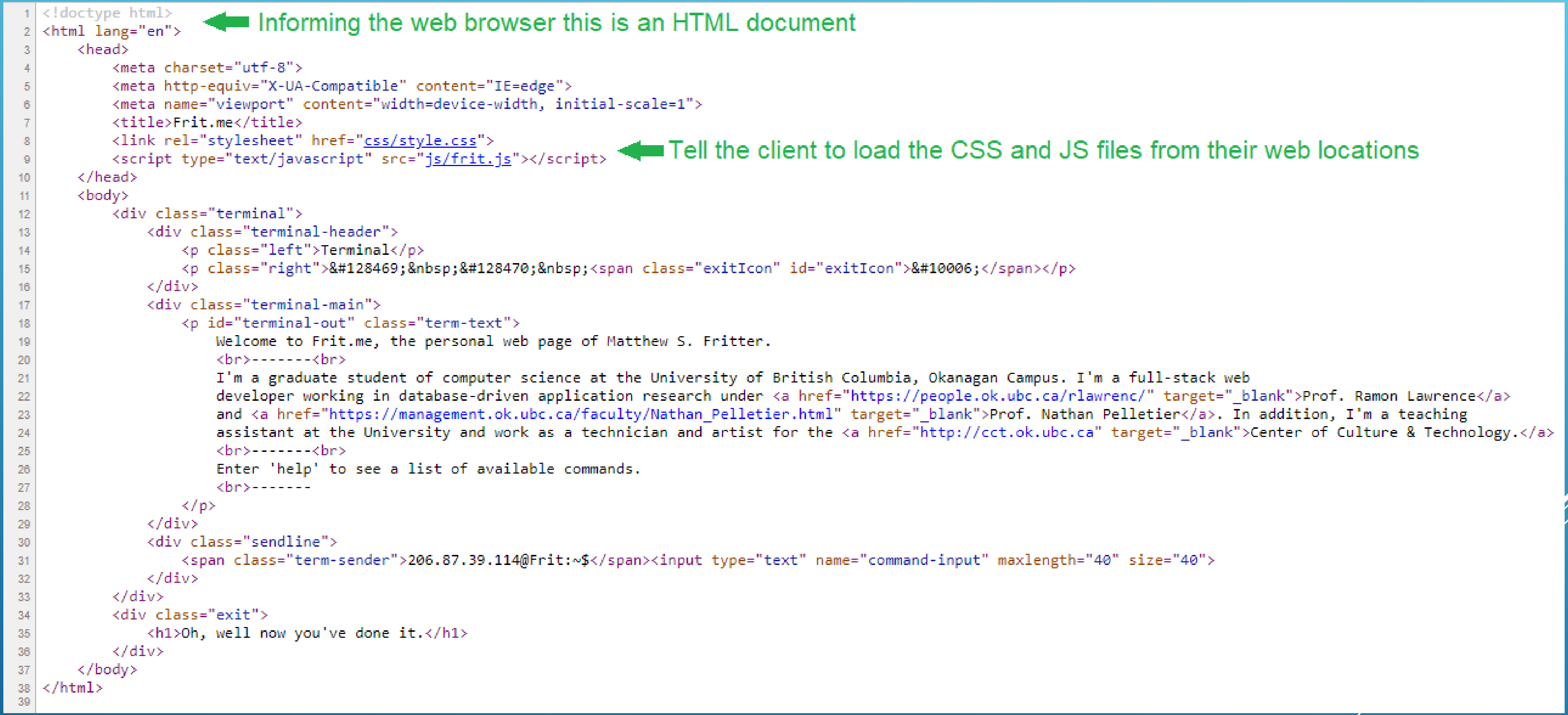
The Web Server

- ***<u>The web server software manages requests made by connected clients</u>***, serving web pages to clients and passing client data to the backend software
- Most web servers also support a variety of back-end languages, either by default, or through additional modules that can be installed. Some web servers are designed from the ground up to support specific languages (Such as Apache Tomcat, which serves Java Server Pages)
- A wide variety of open source and proprietary web server softwares exist. The most popular by market share currently are Apache HTTP Server, Microsoft IIS, and NGINX

The Front-End

- ***The front-end is what you present to clients when they make a request to your website***
- Front-end languages are for the most part defined by what is supported by web browsers. A good front-end needs to adapt to a variety of browsers (including outdated ones), and a number of different devices (cellphone vs. tablet vs. desktop)
- ***Hyper Text Markup Language*** (HTML) provides the "backbone" of a web page, handling text, images, videos, menus, and the other necessary components of the web page
- ***Cascading Style Sheets*** (CSS) provide the styling for the web page, modifying colors, fonts, sizes, and shapes of objects
- ***JavaScript*** (JS) provides client-side scripting capabilities, allowing you to handle interactions, make calls to the web server, and dynamically change the web page in real-time
- Front end web pages may be static, meaning they are hard-coded, or they may be dynamically generated via the back-end of the web service

Example of HTML/CSS/JS Front-end



Informing the web browser this is an HTML document

Tell the client to load the CSS and JS files from their web locations

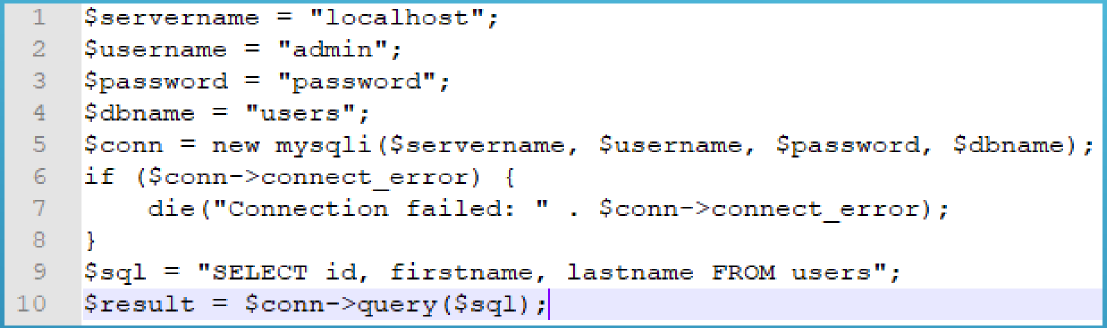
```html
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Frit.me</title>
        <link rel="stylesheet" href="css/style.css">
        <script type="text/javascript" src="js/frit.js"></script>
    </head>
    <body>
        <div class="terminal">
            <div class="terminal-header">
                <p class="left">Terminal</p>
                <p class="right">&#128469;&nbsp;&#128470;&nbsp;<span class="exitIcon" id="exitIcon">&#10006;</span></p>
            </div>
            <div class="terminal-main">
                <p id="terminal-out" class="term-text">
                    Welcome to Frit.me, the personal web page of Matthew S. Fritter.
                    <br>-------<br>
                    I'm a graduate student of computer science at the University of British Columbia, Okanagan Campus. I'm a full-stack web
                    developer working in database-driven application research under <a href="https://people.ok.ubc.ca/rlawrenc/" target="_blank">Prof. Ramon Lawrence</a>
                    and <a href="https://management.ok.ubc.ca/faculty/Nathan_Pelletier.html" target="_blank">Prof. Nathan Pelletier</a>. In addition, I'm a teaching
                    assistant at the University and work as a technician and artist for the <a href="http://cct.ok.ubc.ca" target="_blank">Center of Culture & Technology.</a>
                    <br>-------<br>
                    Enter 'help' to see a list of available commands.
                    <br>-------
                </p>
            </div>
            <div class="sendline">
                <span class="term-sender">206.87.39.114@Frit:~$</span><input type="text" name="command-input" maxlength="40" size="40">
            </div>
        </div>
        <div class="exit">
            <h1>Oh, well now you've done it.</h1>
        </div>
    </body>
</html>
```

The Back-End Language/Framework

- ***The Back-End Language handles the actual logic of the web application***. This includes handling user input (via POST requests), querying and updating the database, and generating front-end output to return to the client
- A variety of languages can be used for back-end processing, including PHP, Java, Python, JavaScript (can be used for both front-end and back-end), Ruby, and C/C++
- Frameworks support web application development by providing a variety of utilities and functions to developers. Many web applications are now developed using frameworks, rather than written from scratch
- The level of additional functionality and customizability vary widely between frameworks, from micro-frameworks that rely heavily on user-written code, to Content Management Systems (CMSs) that are usable by people with little to no programming experience.

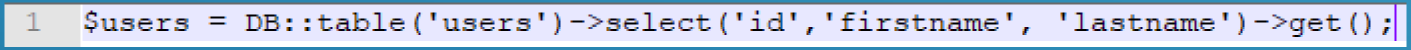
An Example of Framework Functionality

- Below is an SQL query in plain PHP, without the use of a framework:



```php
$servername = "localhost";
$username = "admin";
$password = "password";
$dbname = "users";
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "SELECT id, firstname, lastname FROM users";
$result = $conn->query($sql);
```

- Below is the same SQL query using the Laravel PHP framework:



```php
$users = DB::table('users')->select('id','firstname', 'lastname')->get();
```

The Database

- ***The database serves as a warehouse for our data, providing the functionality required to efficiently store and retrieve large amounts of information.***
- There are a variety of Database Management Systems (DBMS) available, including many free and open-source options
- ***Structure Query Language*** (SQL) based databases are the most widespread, including Microsoft SQL, MySQL, PostgreSQL, and Oracle
- Many databases are installed as stand-alone server software, which can then interface with back-end languages as seen in the previous slide
- Other databases, such as SQLite, may be a single file that is then manipulated by a driver; this allows for lightweight and portable databases when larger databases are unnecessary
- Choice of database software is largely predicated on the requirements of the web application. For many applications, SQLite and MySQL are suitable and free

Our Software Stack

- In this course, we'll be working with a Linux-Apache-MySQL-PHP (LAMP) stack, a common software stack used for web application development
- Specifically, we'll be using:
  - OS: CentOS 7
  - Language/Framework: PHP 7.2 with the Laravel Framework, also the WordPress Content Management System
  - Web Server: Apache
  - Database: MySQL and SQLite

★ In Review

- We’ve now looked at the five major components that make up our software stack:
  - The Operating System serves as the basis for the entirety of the stack
  - The Web Server handles client connections and requests, and supports the back-end language
  - The Front-End is what the web server returns to the client, which is then rendered and shown to the user
  - The Back-End handles the logic of the web application, tying the database, front-end, and web server together
  - The Database allows us to store and retrieve our data in an efficient manner

Questions about the Stack?

Remotely Managing A Server

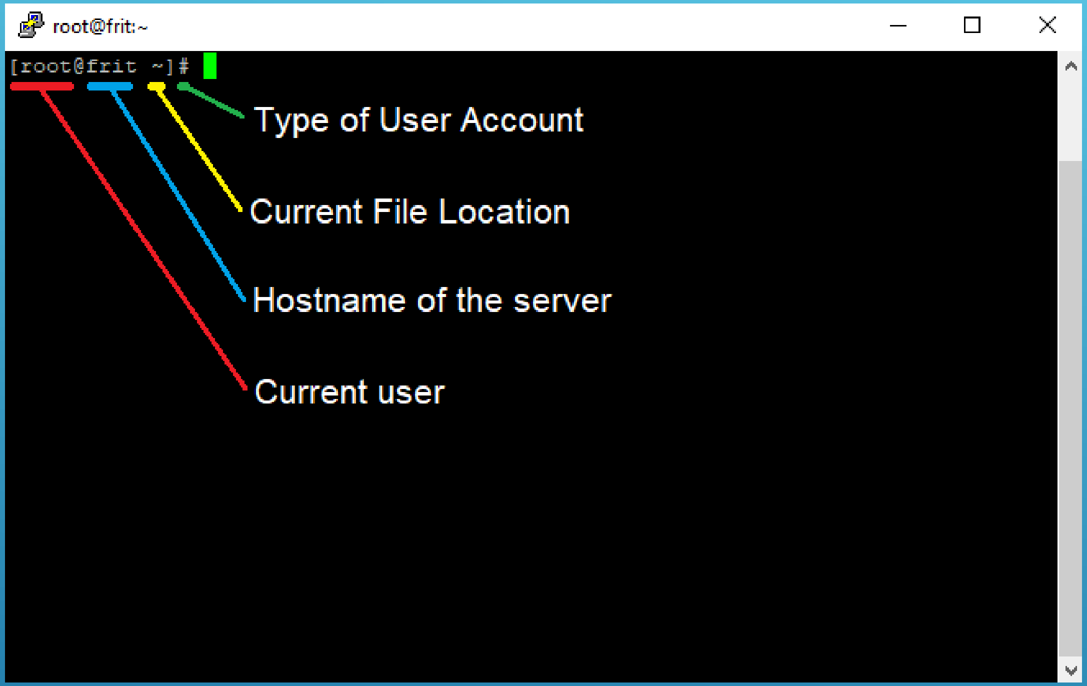
Remote Servers

- With the growing popularity of cloud-based hosting and computing services such as Amazon AWS, Digital Ocean, and Microsoft Azure, a large amount of web development today is done ***remotely***
- Remote servers may be in a data center in a different city, province, or country – in fact, they are likely not physical servers at all, but rather virtual machines provisioned on a shared server
- How then can we set up our stack and manage our web application without physical access to the machine?

★ Secure Shell (SSH)

- ***Secure Shell (SSH) is a cryptographically secure protocol that allows you to remotely login to a computer.***
- By connecting to a remote server using SSH (usually over port 22), we can send terminal commands to the machine. In doing so, we can install, update, and edit files and software on the machine as we please
- This is typically done through terminal on Linux/Mac machines, or by using SSH software such as PuTTY on Windows machines
- In order to use SSH effectively, one must be familiar with some key aspects of the Terminal and a number of useful commands

The Terminal Interface



```
[root@frit ~]#
```

- Type of User Account: # means a root account, $ means a normal shell user
- Current File Location: The folder you are currently in. Note that '~' is an alias for your home folder (unique to your user)
- Hostname of the server: The name given to your server, if any
- Current User: The user you are currently logged in as

★ Important Terminal Commands

Command	Example	Explanation
`cd <location>`	`cd /var`	Change directory to location
`mkdir <name>`	`mkdir myFolder`	Make a new directory with name
`mv <origin> <destination>`	`mv test.txt /var/test.txt`	Move a file or directory from origin to destination
`cp <origin> <destination>`	`cp test.txt test2.txt`	Copy origin file to destination
`cp –R <origin> <destination>`	`cp –R myFolder /var/myFolder2`	Copy origin folder and contents to destination
`ls`	`ls`	List contents of current directory
`ls -l`	`ls -l`	List contents of current directory in list form, showing hidden files and permissions
`man <command>`	`man mkdir`	Shows the manual for a given command
`rm <location>`	`rm /var/test.txt`	Removes (deletes) specified file
`rmdir <location>`	`rmdir myFolder`	Removes specified empty directory
`clear`	`clear`	Clears the terminal window

★ Directories and Locations

- The terminal supports both ***relative*** and ***absolute*** addressing of files and folders
- A relative file path is calculated based on the current directory you are in
- An absolute file path is calculated based on the root directory ('/')
- You can use '..' to refer to the parent directory of the current directory. For example, the command 'cd ..' will take you to the parent directory of the current directory
- Starting a file path with '/' will make it an absolute file path – you must then give each folder and subfolder starting from route to the destination, i.e. '/var/www/html/laravel/.env'
- If a file path does ***not*** start with '/', it is considered a relative file path, based on the current directory

★ The Package Manager

- ***Package Management Systems*** are used in most UNIX-based distros to handle installation and updating of software packages
- Those of you who are familiar with Ubuntu or Debian have most likely used the apt package manager
- CentOS uses the ***yum*** package manager
- To update all packages on your system:

```
yum update
```

- To install a new package:

```
yum install <package name>
```

- To list currently installed packages:

```
yum list installed
```

Additional Repositories

- CentOS by default only pulls from a small repository of available packages – many of which may be older, stable versions of software i.e. PHP 5
- We must install additional repositories if we want to use newer packages i.e. PHP 7
- The two major repositories we will be using are ***Extra Packages for Enterprise Linux (EPEL) and Remi.***
- Example of installing the Remi repository:

```
yum install http://rpms.remirepo.net/enterprise/remi-release-7.rpm
yum-config-manager --enable remi-php72
```

- The above code installs the Remi repository, then enables it as the default repository for installing PHP 7.2

★ Secure File Transfer Protocol (SFTP)

- While it is entirely possible to create, and edit our web files via the Terminal, the limits of console-based editing make it a relatively slow process
  - Terminal editors like ***nano*** are great for making small changes, but lack of syntax highlighting and mouse support make writing large amounts of code relatively painful
  - Other terminal editors like Vi and Vim are more fully featured, but have a steep learning curve
- A more comfortable way to work is to use Secure File Transfer Protocol (SFTP). This protocol, like SSH, uses Port 22, and allows the secure browsing, uploading, and downloading of files
- Using SFTP, we can work in a local environment using an IDE of our choice, then upload our finished work to the server
- There are a variety of free SFTP clients available, FileZilla is a fairly ubiquitous choice and is available for all common platforms (Windows, Mac, most common Linux distributions)

Lab 1 Overview

- We begin our first lab tomorrow (September 12th)
- During this lab, you'll need to use SSH and the package manager to install and configure your server stack components
- Login credentials for the servers will be sent out tonight – I recommend that tomorrow before the lab you try to SSH into your server to ensure your credentials work, and to brush up on your Terminal skills
- You shouldn't need to use SFTP for this lab, so don't worry about it now. Focus on getting comfortable with the shell and using the package manager

That's All, Folks

Any final questions?