COSC 419 – Topics in Computer Science

Fall 2020

So, What is Full Stack Development?

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

  A web development approach in which the developer works on all levels of the software stack

- This kind of development is often associated with Agile and rapid prototyping practices, and it is well suited to solo or small team software development

Why is Full-Stack Development Powerful?

- ***You are in total control of your software ecosystem*** – you can assemble a software stack that meets your requirements and your skills
- ***You are flexible*** – being familiar with the entire software stack, the full stack web developer is capable of filling many roles in an organization
- ***You can be independent*** – full stack web development allows a single developer to build and maintain an entire web application

Full-Stack Developers vs Specialists

- The full stack web developer is a ***generalist***, as opposed to a ***specialist***; full stack web development demands a breadth of knowledge across multiple domains:
  - Systems administration
  - Backend web development
  - Database management
  - Frontend web design
- However, time constraints mean that a specialist will have greater depth of knowledge than the generalist

Elements of the Web Software Stack

- For this course, we’re primarily going to be looking at a traditional web server software stack, composed of:
  - A Linux operating system
  - The Apache web server
  - MySQL/MariaDB/SQLite database
  - PHP/Laravel and Python/Flask web backend logic
  - HTML5/CSS/JS front end

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
- 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

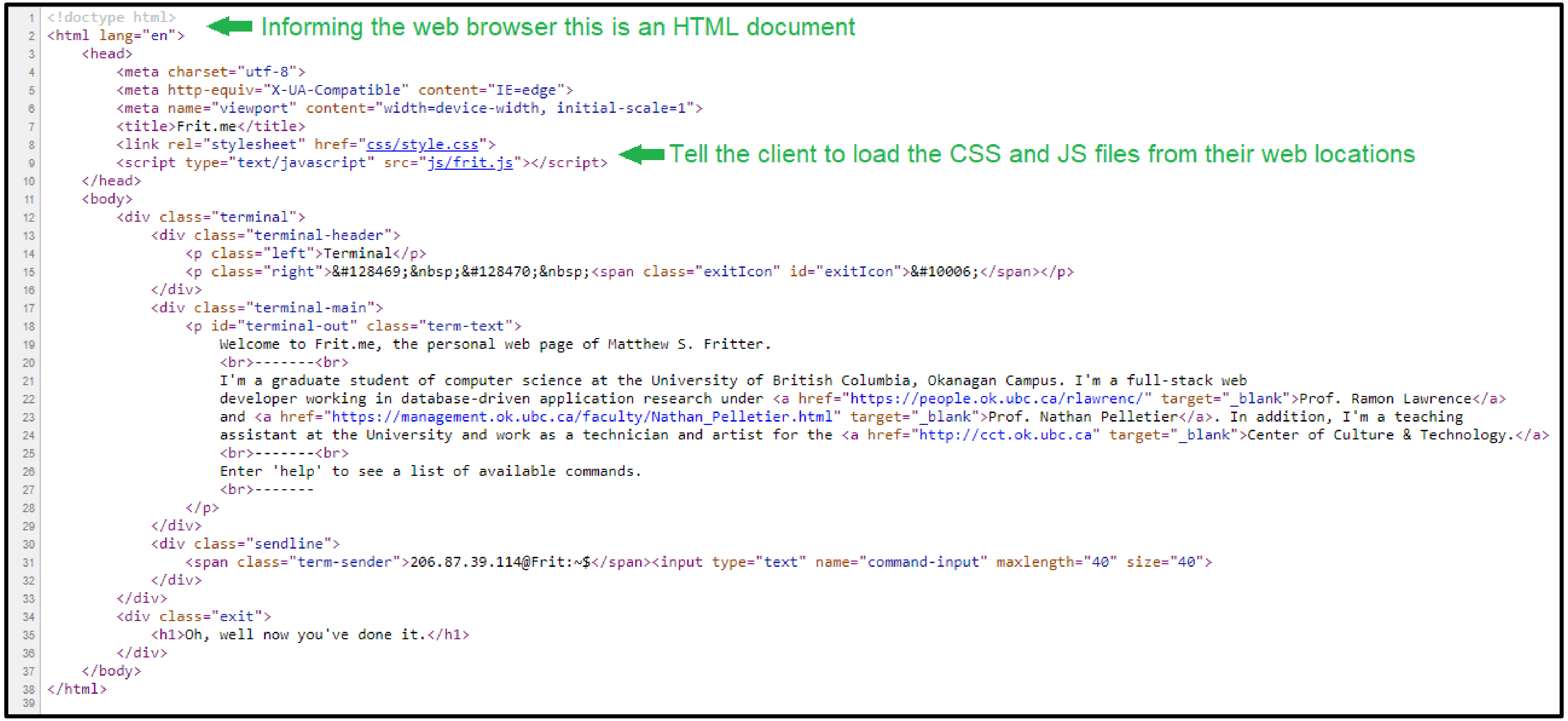
The Web Server

- ***The web server software manages requests made by connected clients***, serving web pages to clients and passing client data to the backend software
- A wide variety of open source and proprietary web server software exists. The most popular by market share currently are Apache HTTP Server, Microsoft IIS, and NGINX
- Additional functionality is provided through modules or packages, providing support for specific backend languages or additional features

The Front-End

- ***The front-end is what you present to clients when they make a request to your website***
- A good front-end needs to adapt to a variety of browsers, and a number of different devices
- ***Hyper Text Markup Language*** (HTML) provides the “backbone” of a web page
- ***Cascading Style Sheets*** (CSS) provide the styling for the web page
- ***JavaScript*** (JS) provides client-side scripting capabilities

Example of HTML/CSS/JS Front-end



```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>
```

Informing the web browser this is an HTML document

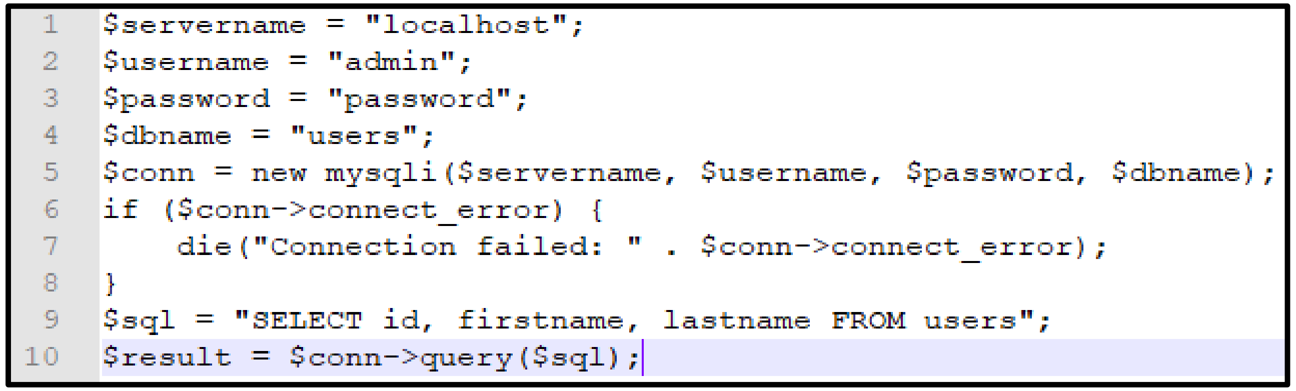
Tell the client to load the CSS and JS files from their web locations

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
- 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

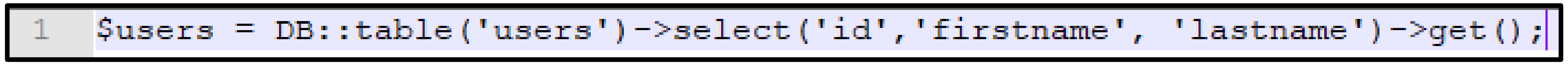
An Example of Framework Functionality

- 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 – we'll be using MySQL, MariaDB, and SQLite
- ***Structured Query Language*** (SQL) based databases are the most widespread, including Microsoft SQL, MySQL, PostgreSQL, and Oracle

Our Virtual Server

- We'll be using a Digital Ocean virtual machine for our virtual servers – these will host our software stack and allow us to serve web traffic from a reliable, fixed IP address
- Will require that we ***connect and work remotely via secure shell (SSH)***
- These remote servers provide:
  - 25GB of disk storage
  - 1GB of RAM
  - 1TB of data transfer per month

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***
- How then can we set up our stack and manage our web application without physical access to the machine?

Secure Shell (SSH)

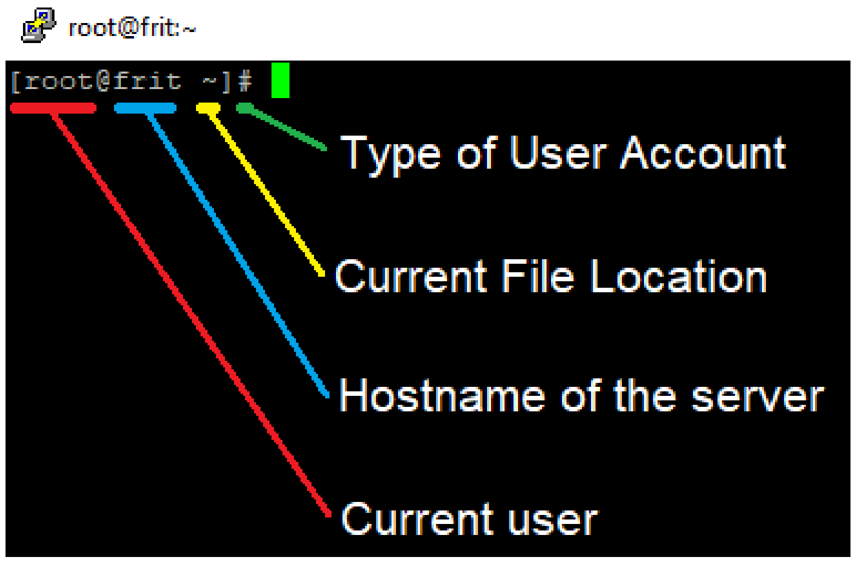
- ***Secure Shell (SSH) is a cryptographically secure protocol that allows you to remotely login to a computer***
- By connecting to a remote server using SSH (usually over port 22), we can remotely control our virtual machine via a command line interface
- In order to use SSH effectively, one must be familiar with some key aspects of the Terminal and a number of useful commands – you will have almost certainly seen these before, but consider this a refresher

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
- 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

The Terminal Interface

- 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



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

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 – relative to current directory, or absolutely referenced 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 root to the destination, i.e. '/var/www/html/app.py'

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: `yum update`
- To install a new package: `yum install <package name>`
- To list installed packages: `yum list installed`

Initial Setup

- When we first provision our virtual machine, there are some configuration steps that we'll want to go through right away:
  1. Update all packages – your virtual service provider may be using an older image that is missing important updates to core packages
  2. Install a decent in-place text editor for quickly changing configuration files without SFTP. The ***nano*** editor is a nice simple option
  3. Create a non-root account with ***sudo*** (**super user do**) access, and then disable non-local root logins

Preview of Lab 1

- Our first lab session is this evening – lab documents will be posted to Moodle/GitHub, and I’ll send out the server login credentials via Moodle
- We’ll be following through with the initial setup steps described in the previous slides, as well as installing our first pieces of software: our Apache web server, and setting up Python 3
- We’ll make a hacky little web application as a basic introduction to Python as well, before we get into Flask in Lab 2

Any Questions?