**File System Navigation**

The command line or Command-Line Interface (CLI) is a way to interact with your computer. As a user, you issue commands to your computer in the form of typing text in the Interface. You can do many things on the command line including:

* Creating folders.
* Creating files.
* Deleting files.
* Deleting folders.
* Moving folders.
* Moving files.
* How to get to your home directory.
* How do you show where you are.
* How do you do auto complete.
* Navigate through and around your folders.

|  |  |  |
| --- | --- | --- |
| **Command's Purpose** | **MS-DOS** | **Linux** |
| Creating folders | mkdir | mkdir |
| Creating files | fsutil file createnew <filename> <size> | touch filename |
| Deleting files. | del filename | rm filename |
| Deleting folders. | mkdir dirname | mkdir dirname |
| Moving folders. | rmdir | rmdir |
| Moving files. | move filename dirname | mv filename dirname |
| How to get to your home directory. | cd \ | cd \ |
| How do you show where you are. | chdir | pwd |
| How do you do auto complete. | tab | tab |
| Navigate through and around your folders. | dir | ls |

**Git Commands**

Git is a source code management system that is widely used in the software development world and performed predominantly on the command line. Using git emphasizes data integrity and speed. With every Git working directory, it’s a full-fledged repository that has full file version-tracking capabilities as well as a complete history. Important actions performed with git commands include:

* Initializing your current working directory as a git repository.
* Adding a remote to your local repository.
* Getting the status of your local repository.
* Stage files for a commit.
* Make a commit.
* Get a list of previous commits for the current local repository you’re in.
* Push your commits to a master
* Push your commits to a specific branch.
* How to pull down changes from another branch and merge them into your local branch
  + You can combine both downloading/fetching changes and merging those changes into your local repository with one git command.
* How do you get a remote branch that you don’t have onto your local computer.
* How do you make a branch.
* How do you push up a local branch.
* How do you delete a local branch.
* How do you delete a remote branch.
* How do you merge a branch into another branch.
* How do you see all your branches.
* Download changes from a branch without auto-merging.

#### Creating Branches

It's important to understand that branches are just pointers to commits. When you create a branch, all Git needs to do is create a new pointer—it doesn’t change the repository in any other way. So, if you start with a repository that looks like this:

Then, you create a branch using the following command:

git branch crazy-experiment

The repository history remains unchanged. All you get is a new pointer to the current commit:

**Heroku - App Deployment**

Heroku is a cloud Platform as a Service (PaaS) that plays well with the git source code management system and Github. Just as you would push your changes to Github, you can push your code to Heroku to instantly deploy your code and have an app ready to go. Important actions to perform when deploying your code to Heroku are:

* Creating a new heroku app for your current local repository right on the command line.
* Pushing/deploying your code to heroku right on the command line:
  + For basic HTML/CSS/JS.

**HTML Basics**

HyperText Markup Language (HTML) is the markup language that gives structure to your webpage. The four (4) core components or elements that consist of a basic webpage are:

* Document type declaration tag or element.
* The actual HTML tag or element.
* The head tag or element.
* The body tag or element.

Within the head tag are two important and meaningful nested tags or elements, specifically:

* Title tag or element.
* Link tags with their associated attributes.

Within the body tag are the actual contents of your web page that you want to show the user/viewer of your web page. There are several important tags or elements to remember, specifically:

* Various sized heading tags or elements (sizes 1 to 6).
* Div tags or elements with their associated attributes.
* Image tag or element their associated attributes.
* Script tag or element their associated attributes.

**Javascript Basics**

Javascript is an interpreted, dynamic, and untyped programming language that is highly flexible. Outside of the web browser, it is standardized as *ECMAScript* and has a specification that you can review to properly apply and write Javascript.

It is one of the three core technologies of the World Wide Web (WWW), which makes viewing websites highly interactive and dynamic. You can create a wide variety of applications, tools and libraries using Javascript.

The important aspects to keep in mind while using Javascript are:

* Declaring variables to capture data.
* Writing/defining functions in two different ways so actions can be performed on data and returning values if desired. These two different ways can be written as:
  + a function declaration (either anonymous or named function).
  + an anonymous function expression (assigning an anonymous function to a variable).
* Properly load Javascript code for a web page either through:
  + VanillaJS (***.onload()***) or
  + jQuery (***.ready()***)
  + Why one is better than the other.
* Using ways to store data of various types (“strings”, numbers, boolean - true/false):
  + Using arrays to store data, which are essentially an ordered lists.
  + Using Javascript objects to store data, which are basically unordered lists.
  + Using arrays to store Javascript objects so you can store more data in a single go.
* Creating flow control to execute specific lines of code according to conditions such as:
  + Using if...else statements to execute pieces of code if a specified condition is evaluated to **true**.
  + Using switch...case statements to evaluate an expression and matching that expression’s value to a **case** clause and executing statements associated with that **case**.
  + Use comparison operators to compare values which evaluate to true, such as:
    - Strict Equal (===).
    - String note equal (!==).
    - Great Than or Greater Than or Equal To (> or >=).
    - Less Than or Less Than or Equal To (< or <=).
* Use arithmetic operators to perform simple arithmetic. These arithmetic operators are:
  + ( + ) Addition between **Number** data types.
  + ( - ) Subtraction between **Number** data types.
  + ( \* ) Multiplication of **Number** data types.
  + ( / ) Division of **Number** data types.
  + ( %) Modulus which return the remainder of a Division operation between **Number** data types.
  + ( ++ ) Incrementor
  + ( -- ) Decrementor
* Using iterators to loop through arrays to access elements containing pieces of data.
* Creating/using VanillaJS (plain Javascript) event listeners to allow interaction with a web page.
  + using jQuery as an external library to more easily create event listeners such as a click or hover event.
* Using VanillaJS (plain Javascript) to traverse the Document Object Model (DOM).
  + Using jQuery as an external library to easily traverse the DOM.
* Using Vanilla JS (plain Javascript) to manipulate the DOM.
  + Using jQuery as an external library to manipulate the DOM.
* Using Vanilla JS (plain Javascript) to apply attributes and styling to DOM elements.
  + Using jQuery as an external library to more easily apply attributes and styling to DOM elements.
* Using callback functions to perform asynchronous tasks. These callbacks can be in the form of:
  + Anonymous functions.
  + Function expressions (assigning variables to function declarations).

**Application Programming Interface - Consuming**

An Application Programming Interface (API) is a structured set of tools, routines, and protocols for software to interact with other pieces of software. To consume an API means to access essentially another database with stores of data.

The most common way to consume an API is using the external Javascript library, jQuery. The most often used of the jQuery methods which allow you to make an asynchronous HTTP request:

* jQuery.ajax() **OR** $.ajax()

**Firebase**

Firebase is a backend as a service and cloud services provider that provides a realtime database through an API, specifically a REST API, that allows application data to be synchronized across clients (mobile, tablets, web) that is stored on Firebases’s cloud. Firebase also acts as a user authentication service using only client-side code.

* Create a reference to a **Firebase** database so we can store data.
* Save data to our **Firebase** database using **.push()**.
* Listen to your **Firebase** database when your data changes with **.on(“*value*”, callBack)**.
* Listen to your **Firebase** database when a *new child* is added with **.on(“*child\_added*”, callBack).**
* Display your **Firebase** data using **jQuery** in the DOM.