Notes

This document is where I will be typing my notes as I do the pre-work for this summer's Swift curriculum.

Technical Vocabulary

The following terms appeared in a lesson as important vocabulary.

- **argument**:When you call a function that has an argument, it means that the function needs some sort of value to run. For example I created the following function and the argument here is the Int after numberOfPencils.
  - `func colorPencils(numberOfPencils : Int) {...`
- **array**: an ordered collection of stored values under one variable
- **call/run**: telling a function to do its specific job
- **camelCase**: when a variable is named something that is multiple words, lowercase for the first word and capitalize the first letter of the following word(s)
  - var **lastName** = "Williams"
- **collection**: a way to hold multiple pieces of data
- **constant**: defined by `let` a constant is a variable that will never change
  - `let firstName = "Ashanti"`
- **declare**: assigning a function a specific ob
- **dictionary**: dictionaries are ways to hold data that needs to be associated with another piece of data; think a word with its definition
- **double**: a number that has a decimal, cannot be combined with an integer/float
- **element**: numbers, strings, arrays, etc; whatever the array holds
- **function**: an action in the code
- **index**: the number automatically assigned to items in an array based on its position, the first item in an array will be indexed at 0
- **initialize**: how to create an array or dictionary
- **integer**: a number without a decimal, cannot be combined with a double (sometimes referred to as a float)
- **interpolation**: including variable information in an output
  - ```
    var listName = "Reminders"
    var totalReminders = "five"

    print("There are \(totalReminders) items on the \(listName) list.")
    //=> There are five items on the Reminders list.
    ```
- **key/value pair**: the key is the word and the value is the definition; key is the label and value is the data
- **keyword**: special reserved words in a language
- **loop**: when code runs through multiple iterations of itself
    ```

- **parameter**: parameters are the information located in the parenthesis immediately following the name of a function
- **return value**: the value that is the result of a function being run. It is stored for use later on in the code.
- **string**: a series of characters between two double quote marks
  - `"hello!"`
- **unwrap**: unwrapping tells Swift that you are sure there is data present in a dictionary and you want that data returned
- **variable**: defined by `var` followed by the name of the variable and what the variable is
  - `var name = "Ashanti"`

Git

Pushing to Git

1. `git status` - check to see what files have changed
2. `git add .` - adds changes to the directory
3. `git status` - checks to make sure the files you added were staged (should turn green in terminal)
4. `git commit -m "Some Message"` - adds a commit message
5. `git push` - pushes the changes to github

Things to Remember

- **branches** - sometimes it's better to work on a different branch outside of the master branch
  - `git checkout -b nameOfBranch` - creates a new branch
  - `git checkout nameOfBranch` - moves you to a different branch
  - `git merge nameOfNonMasterBranch` - will merge the master branch (assuming you are in the master branch) with the changes created in a different branch
  - `git branch -d nameOfBranch` - deletes a branch
- the git status in between steps aren't necessary but they are nice to look at every now and again. they can save you from making mistakes and pushing something on the master that you didn't intend to or force you to save something that didn't save for a certain commit

Commands

Command	Function	Note(s)
`!=`	comparison operator - not equal	boolean - will return a true/false
`%`	finds the remainder of two numbers when they are divided	
`<=`	comparison operator - less than or equal to	boolean - will return a true/false
`<`	comparison operator - less than	boolean - will return a true/ false

Command	Function	Note(s)
`==`	comparison operator - equal	boolean - will return a true/false
`>=`	comparison operator - greater than or equal to	boolean - will return a true/false
`>`	comparison operator - greater than	boolean - will return a true/false
`Int.random()`	picks a random number	can be used to pick a random item in an array
`\()`	used to interpolate variable information in an output	
`for...in`	a loop that will run code a specific number of times, can be used to iterate over a collection	
`func`	declares a function	
`let [name of constant] =`	sets a constant (unchanging variable) equal to something	
`nameOfArray.append()`	adds something to an already existing array	
`nameOfArray.count`	returns the number of elements stored in a particular array	
`nameOfArray.remove(at: #)`	replace the # with the index number of whatever you want to remove from the array	
`nameOfDictionary.removeValueForKey("nameofKey")`	removes a specific key from a dictionary	
`nameOfDictionary["new key"] = "newValue"`	adds a new key-value pair to a dictionary	
`print()`	prints something to the console	

Command	Function	Note(s)
`print(nameOfDictionary["keyInDictionary"]!)`	prints the value associated with the key you selected	the `!` is used to unwrap the value, if it is left off, the data will print with an optiona
`return`	returns a value from a function; also terminates the function	
`var [name of variable] =`	sets a variable equal to something	
`var nameOfArray = [String]()`	initializes an empty array that will contain strings	can replace `[String]` with whatever you want in the array
`var nameOfDictionary : [String : String] = [:]`	initializes an empty dictionary	can replace `String : String` with whatever you want in the dictionary