Intro to Programming: Summary notes

Overview:

- **Code:** set of special instructions to tell the computer which tasks to perform.
- **Syntax**: rules of valid format and instructions, also known as the computer language

Statements: a group of words, numbers, and operators that perform a specific task.
- example: a = b * 5;
- 'a' and 'b' in this example are variables
    - variable = a named location stored in memory that is used to hold a value which can be modified by the program.
- '5' in this example is a literal value
- the '=' and '*' are operators (they perform actions with the values and variables)
- most statements end with a semicolon ;
- programs are collections of many statements, which together describe all the steps it takes to perform your program's purpose

Expressions: any reference to a variable or value or set of variables and values combined with operators
- 'call expression' statement (example: alert (a);

Executing the program: typically done from top to bottom, line by line, every time the program is run. This is known as "interpreting the code"

Output: how we print text, example of how to create output is console.log();
- you can also use alert(); which will show output as a popup

I**nput:** receiving information from the user
- examples: an HTML form or using the prompt() function

Operators: how we perform actions on variables and values
- There are many operators in javascript
- some examples are =(assignment), math (+, -, * , / etc)
- Compound Assignment: combine assignment and math operators (+=): a+=2 would mean (a = a+2)
- Object property access (.) as in console.log
- Equality operators (==, ===, !=, !===)
- Comparison (<, >, <=, >=)
- Logical (&&, II,) (and, or)

Values and Types:

- Number (when you need to do math)
- String (one or more characters, words, or sentences)
- Boolean (when you need to make a decision)
- Literals (values that are included directly in the source code)

Converting Between Types:

- coercion (converting a string to a number and vice versa)
- Number() - a built in function that would convert any type to a number
- Javascript sometimes uses implicit coercion

Code Comments:

- How you write code matters
- it is important for others to understand and clearly read your code
- Shouldn't need to overuse comments, as it is a sign of poorly written code
- Comments should explain Why, not What

Variables:

- a symbolic container that is assigned values
- javascript uses "dynamic typing" (variables can hold values of any type without type enforcement
- declare a variable using *var*
- console.log(amount.toFixed(2)); (would use the variable amount and set the result to 2 decimal places)
- a *constant* is a variable that does not change throughout the program
    - you can use *const* to declare the constant variable

Blocks:

- grouping a series of statements
- wrapping one or more statements inside a curly brace {}
- often combined with an if statement or a loop
- a block statement does not need a semicolon ; to end it

Conditionals:

- decisions
- *if* statement (if this condition is true, do the following…)
    - requires an expression in between the parenthesis() that can be treated as true or false
- if the condition isn't true, you can provide an *else* clause
- *switch* statements (a series of if…else statements)
- *loops* use a conditional to determine if the loop should keep going or stop

Loops:

- repeating a set of actions until the condition fails
- includes the test condition and a block
- *iteration (*each time the loop block runs)
- the while loop and the do..while loop repeat a block of statements until a condition no longer is true
- The only practical difference between these loops is whether the conditional is tested before the first iteration (while) or after the first iteration (do..while).
- the *for* loop has 3 clauses: initialization (var i=0), conditional (i<=10), and the update clause (i = i+1)
    - the *for* loop is good for counting

Functions:

- a named section of code that can be “called” by name, and the code inside of the function will run each time
- can take arguments (parameters), aka values you pass in.
- can return a value back
- often used for code that you plan to call multiple times
    - can also be useful just to organize related bits of code into named collections even if you only use them once

Scope:

- a collection of variables and the rules for how those variables can be accessed by name
- only code inside that function can access that function’s *scoped* variables

Power of Practice

- the best way to learn how to write code is to Write Code