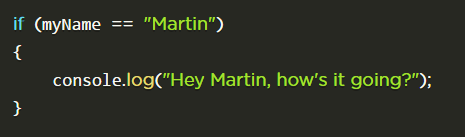
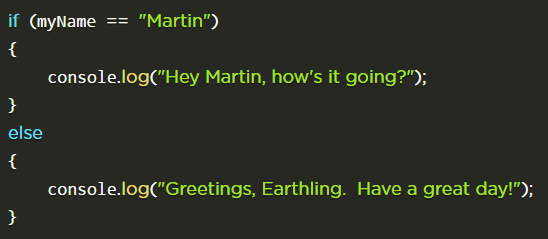
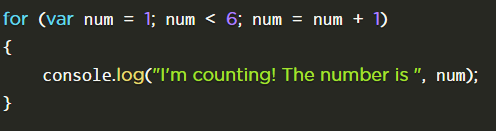
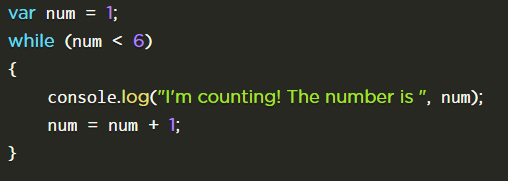
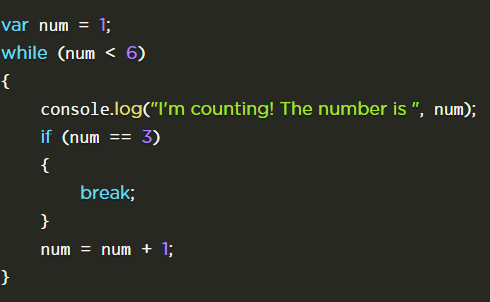
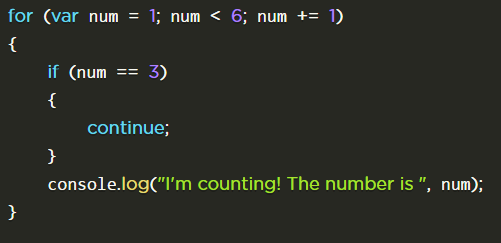
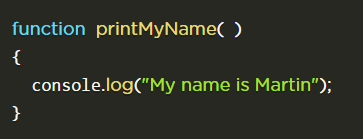
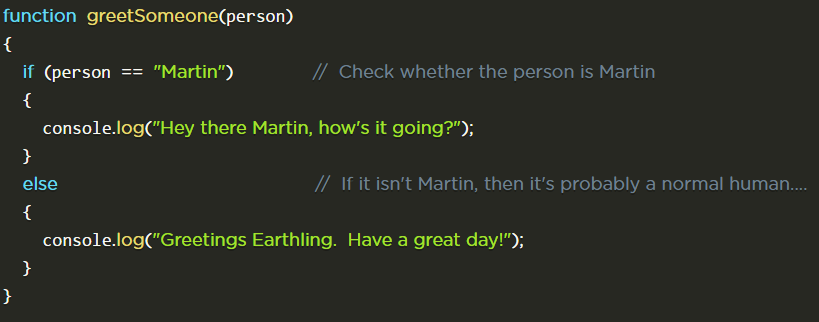
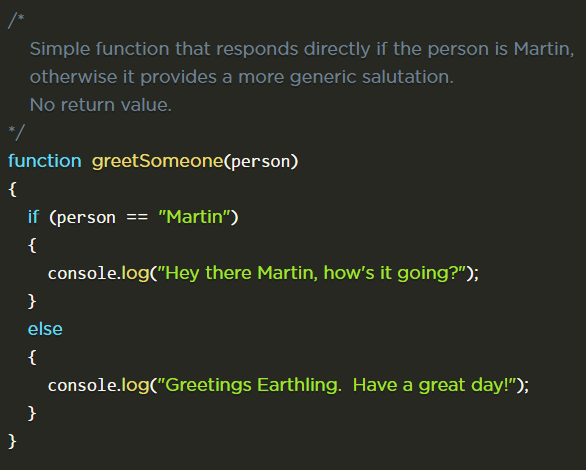
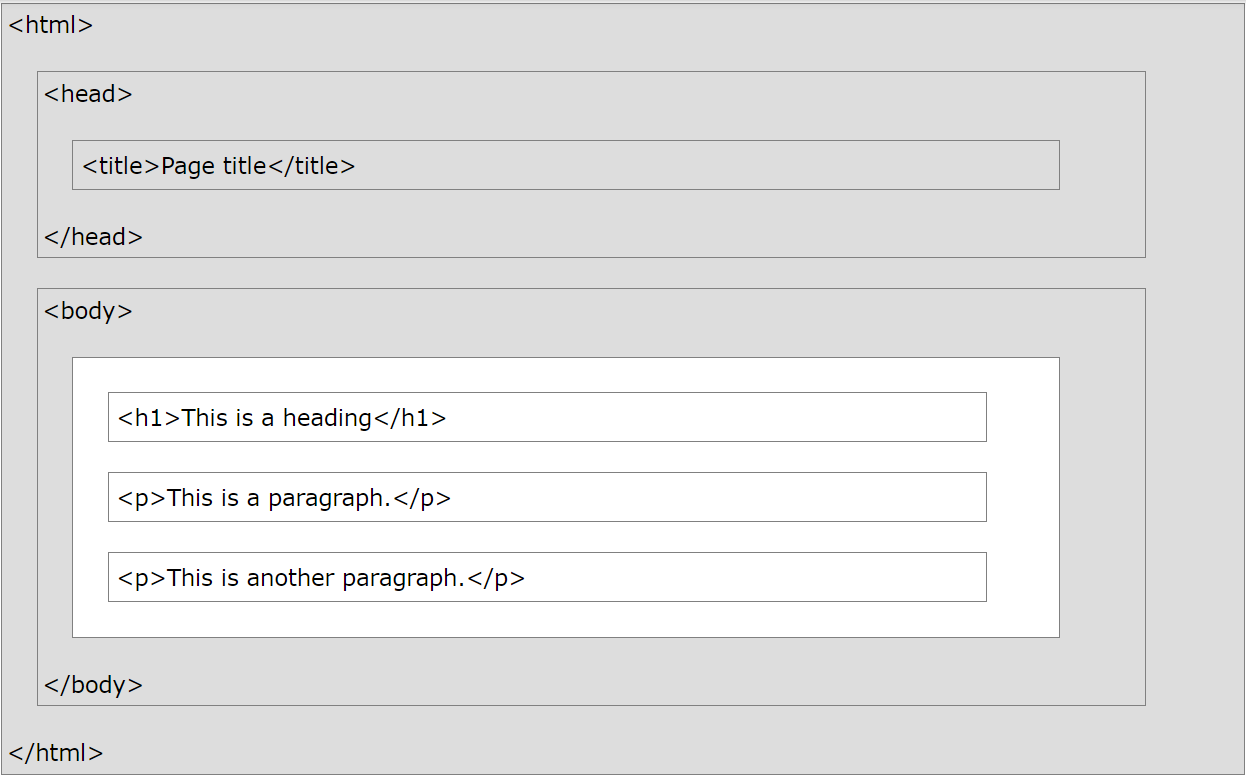
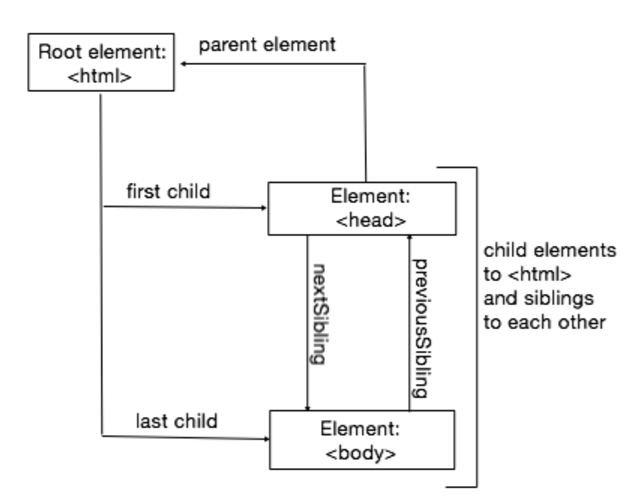
**Coding Bootcamp Notes**

* What is **source code?**
  + Is basically coding, it is what the developer or engineer writes down a series of instruction that the computer than reads and acts upon
  + b/c of this the **code needs to flow**, it needs to move in a linear fashion
    - you can instruct a computer to execute a section of code multiple times which is called **Loop**.
    - code that you expect to use often, whether called by various places in your own code, or perhaps even called by others' code, you can separate this out and give it a specific label so that it can be called directly. This is called a **Function**.
  + **Console.log(“Hello World!”);**
  + Tells the computer to print out “Hello World!” to be displayed
  + To log a message to the console, we will use the console.log( ); command. Between the parentheses, we can put any message we want to be displayed
* **Variables**
  + A variable is a specific spot in memory, with a label that you give it. You can put anything you want into that memory location and later refer to the value of that memory, by using the label.
    - For example: **var myName = ‘Julie’;**
      * Now we know a variable with the label ‘myName’ which = the value ‘Julie’.
      * So when we want to reference it we can just call ‘myName’
  + a **string** is traditionally a sequence of characters, either as a literal constant or as some kind of variable. The latter may allow its elements to be mutated and the length changed, or it may be fixed (after creation).
* **The difference between ‘=’ and ‘==’ and ‘===’**
  + **=** sets things
  + **==** compares things
  + **===** is stricter in that the variables MUST equal each other
  + **!==** means they aren’t comparable or Does not equal each other
  + **<= or =>** “equal to or less/more than” comparisons
  + **++ or += or num = num+1** all mean the same thing. It’s just telling you to go up by 1.
* **Conditionals: If and Else Statements**
  + each decision has only two possible outcomes -- you have a certain test or comparison that you do, and **IF** the test passes then you execute certain code.
    - 
  + We can also use the **ELSE**  statement if we want something to happen if the above comparison didn’t happen
    - “If user is not named MARTIN then run this string” AKA **ELSE**
    - 
  + You can create compound tests, using a combination of the logical AND, OR and NOT connectors just like you might in natural language.
    - **II** we designate the logical **OR** comparison with a pair of vertical bars
    - **EXAMPLE:** *"if it is raining or if it is too far to walk, let's call Uber!"*
      * If(raining == true II distanceMiles>2)
      * {
        + callUber();
      * }
  + possible to deepen the control flow of an IF..ELSE statement by making it incorporate additional IF..ELSE statements within it. 🡪**"nesting"** or **"chaining"** IF..ELSE statements.
* **Loops**
  + essentially a section of code that will be executed a certain number of times
    - **FOR** loops are good when you know exactly how many times those lines of code should be run.
      * **FOR loop**, in addition to the chunk of code to be looped, you need to specify three things:
        + 1) any initial setup,
        + 2) a test that needs to be true to continue looping, and
        + 3) any code that should be run at the end of each time through the loop.
      * 
        + it would continue to log until it hits 5 in which the conditions can’t be met since variable num is not less than 6.
    - **WHILE** loops are a combination of loops with conditionals; handy if you don't know ahead of time how many times your program should run through a block of code, but you do know that you want the program to run through that block of code repeatedly for as long as the condition for the loop will remain true.
      * doesn't automatically include code that is always executed at the end of each loop
      * also, the FOR loop above could be rewritten as:
      * 
  + If you add **BREAK**; to your code, program execution will immediately exit the specific loop that you are currently in, and continue execution immediately following the loop.
  + 
    - It will log 1 and 2 but at 3 it will skip out of the loop and not log anymore.
  + If you add **CONTINUE**; to your code, the rest of that pass through the loop will be skipped but the loop-end statement is executed and looping will continue.
  + 
* **ARRAYS AhhhhYAYEAHHH**
  + An array is like a cabinet with multiple drawers, where each drawer can store some information such as a number, string, or even another array/cabinet.
  + And it is always signified with  **[]**
    - **Example**: var x = [1,3,5]
  + *The array first position is always a* ***0 and then goes 1, 2,…n***
  + The **push()**  function just tacks on a number to the end of the array
    - The **pop()** takes out a number at the end of the array or whatever is written between the ().
* **Functions**
  + **Function:** code that you expect to call often, you can separate this out into a different part of your source code file, so that these lines of code don't need to be duplicated each time you want to print out something.
    - 
    - this does not actually call the function immediately; it only sets it up for other lines of code to call the function later.
      * also, ‘calling’ the function is also called 'running' or 'executing'
* **Parameters**
  + you can pass values into functions, so that the functions can behave differently depending on those values. The caller simply inserts these values (called **arguments**) between the parentheses, when it executes the function.
  + **{}** braces are a way to indicate to the system some number of lines of code that it should treat as a single group. Without these, IF..ELSE and WHILE and FOR statements will only operate on a single line of code.
* **Return Values**
  + Functions (usually) have names. They (often) have parameters. They have code that will run when the function is executed. They generally have a return value as well, which is simply a value that is returned back to the caller when the function finishes executing.
    - once the return statement runs, any subsequent lines of code in our function will not be executed. **When program execution encounters a RETURN it exits the current function immediately.**
* **Comments**
  + **<! ….>** also adds comments and isn’t read by the program.
  + **//**
    - 
  + **/\*…\*/**
    - can span multiple lines, and everything between them is considered a non-source-code comment.
    - 

**How the Web Works**

* **Clients** are the typical Web user's Internet-connected devices (for example, the computer you are on right now connected to your Wi-Fi!) and web browsers on those devices (like Chrome or Firefox)
* **Servers** are computers that store web pages or applications
  + the client makes a **request** and the server answers back with a **response**.
    - Web server🡪language🡪database and then returns information
* A **URL** is Uniform Resource Locator
  + **IP** **address** is simply**one computer's unique identifying address.**
    - we point the IP address to a domain name. When you type in www.cnn.com, your web browser visits something called the DNS (Domain Name Server), which acts like a phone book.
* (Client🡪DNS🡪Servers)=Requesting something
  + Going the opposite way is called “sending a response”

**HTML**

* “Hyper Text Markup Language”
  + Let’s us communicate with the browser on what we want to put on it.
    - To do this it assigns “TAGS” to wrap around the content w
      * Headings and paragraphs
      * Images
      * Links
      * Lists
      * Tables
      * Forms
    - This essentially describes the content that they contain
      * The browser doesn’t display the HTML tags but uses them to interpret the content of the page
* Opening and Closing Tags
  + Most tags come in pairs since they are supposed to wrap things.
    - The first tag in a par is the “start tag” and the second is the “end tag” or aka “opening” and “closing”.
    - The end tag is written like the start tag but with **a forward slash inserted before the tag name** **<p>….</p>**
  + 
    - The first line tells the browser that we want the document to be read in standard mode “we are using the latest accepted HTML conventions”
      * This should always be the first line in every HTML doc.
    - The 2nd line indicates that everything between it and its closing tag will be marked with HTML conventions.
    - The 3rd line will eventually contain meta data
      * Meta data is data
    - The 4th line contains user content🡪everything that will be visible in the browser
    - The **<div>** is used for additional sectioning (division)
* The HTML Page Structure
* 
* **Parent, child, sibling relationship: PCS**
  + PCS: is a term to describe the relationship between tags/elements inside an html document.
  + 
  + 
    - <html> is the root element, it has no parents
      * It is the parents of <head> and <body>
      * <head> is the first child of <html>
        + <head> has one child: <title>

<title> has one child which is a text: “hello world!”

* + - * <body> is the last child of <html>
        + <body> has two children: <h1> and <p>

<h1> has one child: “Here is a heading tag”

<p> has one child: “Now a paragraph tag”

* + - * + <h1> and <p> are siblings
      * <head> and <body> are siblings.
* **Head**
  + 
  + <head> is the **opening tag** which indicates the properties of the document.
  + **Meta Tags** *description*attribute describes the basic purpose of your web page **( a summary of what the web page contains)**. For each web page, you should place a concise and relevant summary inside the meta *description*tag
    - **<meta charset= “utf-8”>**
      * that meta tag **basically specifies what character set is your website written with.** ... UTF-8 (U from Universal Character Set + Transformation Format—8-bit) is a character encoding capable of encoding all possible characters (called code points) in Unicode. The encoding is variable-length and uses 8-bit code units.
* 
* **<a>** defines the html links
  + **<img>** is the image tag

**Nested HTML Elements**

* Elements can contain elements
  + This is defined as the PCS or the Parent, Child, Sibling relationship