* HTML (hypertext markup language): specify content and their structure
  + Paragraphs, headings, sections
* CSS (cascading style sheets): specify how html is presented, gives the page style separately from the structure.
  + Colors, fonts, borders, margins, layout
* JavaScript: creates dynamic behavior in web pages (compute, react, draw, communicate, alert, alter, update, change, etc)
  + Check user’s form input, grab tweets and display them, etc.
* DOM (document object model): how browser rep. webpage internally
  + Webpage creates DOM based off HTML
  + HTML = structure NOT presentation
  + Upside down “tree of objects”
  + Nested structure of HTML is the same as the nested structure of the html
    - Elements in DOM are used by JavaScript
    - document.getElementById(“ “);
      * add id to HTML element to use JavaScript to manipulate
      * h1.innerHTML = “New Heading!”;
        + modify the page by modifying the DOM
* Core Language: array, math, date
  + Language that is the same however/wherever you are using JavaScript
* User objects: objects you make yourself
  + The way you create objects is the same, but you will make them to fit your needs
* Host objects: specific to the environment
  + Document, window, history, h1, h2, form
* Basic code writing: Write code, compile, execute
  + Compiled languages are strict with how you code, but fast to execute
* Javascript can deploy code with any page using <script> tag
  + “scripting” language: less strict, more forgiving, interpreted without a forming compilation stage
  + Javascript is now faster! Flexibility and speed
* JS: core language and how to make use of the environment supplied by the browser
  + Core: parts of the lang that lets you do basic, computational things
  + Environment: get user input, draw graphics, location, store data, etc.
* JS is not only in a browser: Photoshop, MAX6, OpenOffice, etc.
* JS is easy to get started with!
  + Official programming language of the Web
  + Simple computation is easily to describe
  + Strings, variables, functions
    - Just need text editor and a browser
* JS = describing computation
  + Step by step problem solving
* Writing a statement
* Variables and Values
  + Variable declarations: usually all at the top of the code to keep track of what you’re using (and others to understand your code)
  + Variable: a name you put on storage to add value
    - Values can be numbers or characters and be used to test value, change value, or compute new values
  + Ex: var name = “Fido”; (string)

var age = “4”; (number)

* + - Numbers can be decimals as well.
      * In other languages, decimals are floating point numbers and whole numbers are integers
  + Types of variables: JS is dynamically typed
    - Can reassign a variable to another type
    - Java = variable has to be declared and stay with the type
      * Static language
      * Can figure out if you can make errors when compiling; can lead to fewer runtime errors
    - Changing type in the middle of a program can lead to unexpected behavior, so use with care!
  + Primitive: null, undefined, Boolean, number, strign
  + Objects: array, document, objects you create yourself

Doing things more than once

How to make decisions

Communicate with user

* + Alerts
  + Directly write in document
  + Print to Console
  + Manipulate DOM
* Coding a “serious” JS app

<script>

* Ch. 1 pg. 11-13 (values and variables)
* Variables are used to store values
  + Numbers, strings, Boolean
    - Start your variables with a letter, underscore, or $
    - You can use as many letters, digits, underscores, or $ as you like
    - Don’t use built-in keywords & JS is case-sensitive
    - Choose names that mean something to your code
    - Use camel case when you creating multiword variable names
    - Use variable that begin with \_ and $ only with very good reason
      * Stay away as much as possible
  + Syntax rules
    - Each statement ends in ;
    - A single line comment begins with //
    - Whitespace doesn’t matter
    - Surround strings with double quotes
      * If using single quotes, just be consistent throughout entire code
    - Don’t use quotes around Boolean true and false
    - Variables don’t have to be given a value when they are declared
    - JS is case sensitive
      * Counter will be different than counter
  + Quick Activity Pg. 14 and 39

A: inconsistent quote, quote around Boolean False, $before punchline, no semicolon after string, capatilized Alert, no semicolon or equal for result

B: space between zip code, apostrophe in variable name, quotes missing around string

* + - A results: missed % for variable, cannot use. Missing open bracket in if, else command.
    - B results: forward slashes backward on opening comment. If, else doesn’t work because of apostrophe in variable name
* Ch. 2 pg. 55 (operators)
  + Boolean operators: comparison and logical
  + Boolean expressions result in true or false values
    - Comparison: Compares two values
      * < less than, > greater than, == equal to, === exactly equal to, <= less than or equal to, >= greater than or equal to, != not equal to
    - Logical: combines two Boolean expressions for one Boolean result (T or F)
      * || OR, && AND, ! NOT
* Ch. 3 pg. 98-111 (global vs. local)
  + Variable scope: if it is declared outside of a function, it can be used anywhere in your code.

If a variable is declared inside a function, then you can only use it within that function.

* + Two kinds of scope:
    - Global: declared outside a function and can be accessible anywhere in your code.

var avatar;

var levelThreshold = 1000;

* + - Local: declared inside a function and only accessible within that function. In the example below, the variables i, score, and points are all local. We can call back the variable levelThreshold because it is global.

function getScore(points) {

var score;

var i = 0;

while (i < levelThreshold) {

//code here

i = i + 1;

}

return score;

}

* + Using the variable i is an old practice that has stuck and transcended the “meaningful name” rule of variables.
    - Programmers use it for the variable you iterate with.
  + Globals live as long as the page (refresh=destroy)
  + Locals typically disappear when your function ends
    - Even if your variable appears for the first time in a function, not declaring it with “var” first makes it global.
    - It behaves as if you declared var x = o; at the beginning of the function, making it global.
    - Can put the global variable in the shadow of the local if it is declared outside and inside of the function
      * If you change one, it will have no effect on the other. They are independent variables.
  + Global variables are easier to locate and keep track of by outing them at the top
  + Keeping functions grouped together, but not necessarily at the top, makes it easier as well
  + Declare local variables at the beginning of the function body
* Ch. 1 pg. 11-13 (variables and values)
* Quick Activity Pg. 14 and 39
* Ch. 2 pg. 55 (numbers, strings, Booleans)
* Ch. 3 pg. 98-111 (global vs. local)
* Chapter 1 - Pages 11-13 (Variables & Values); Quick Activity & Solution on pages 14 & 39;
* Chapter 2 - Page 55 (Numbers, Strings, Booleans)
* Chapter 3 - Pages 98-111 (Global vs. Local)
* Chapter 1:  (Conditionals - Making Decisions) Pages 17-23
* Supplement Reference:  Chapter 2 (Looping) Pages 51-55
* Chapter 3: 81-103 (videos)
* Chapter 10: 430-462 (video)
* Chapter 11: 476-506 (video)
* Chapter 4: 126-161 (video)
* Chapter 7: 265-316
* Chapter 2: pg.55