Resources from Ivan Turner, BTHS:

<https://github.com/gnrlwoundwort/Web-Development>

HTML Documents

* Homepage file is named index.html
* Use lowercase when possible
* Start with <!DOCTYPE HTML>
* Comments go between <!-- and -->. Comments do not execute
* Whitespace doesn’t matter
* Siblings: same level of indent
* Child: one level indent from parent
* Tags/HTML elements
  + All tags are objects
    - Objects are referred to left of a dot
  + Goes between <and>
  + Closing tag starts with </
  + Backend code
    - the code goes between <html></html>
    - String between <title></title> shows as tab title
    - CSS (for design) goes between <style></style>
    - JavaScripts go between <script></script>
    - Code runs on the page
  + Front end code
    - Goes between <body>and</body>
    - Code displayed on interface
    - Main text on the page
    - </br> create a line break
    - Paragraphs go between <p>and</p>
    - Block element: element takes the whole line, but resizable
      * String between <div>and</div> gets its own line
        + Start with <div id = “name”> to name the object
    - Inline blocks: takes up the space they need like inline, but resizable like blocks
    - Inline elements: take up only the space of their content
      * String between <span>and</span> goes on the same line as adjacent spans, with a space between the strings.
        + Start with <span id = “name”> to name the object
    - Headings
      * Goes from <h1></h1> to <h6></h6>
      * As number increases, font size decreases
    - Formatting
      * Bold strings go between <b>and</b>
      * Italic strings go between <i>and</i>
    - Buttons
      * <button onclick = “function”>name</button>
    - Images: <img/>
      * src = “relative path” is source
    - HTML Entities
      * Starts with & and ends with ;
      * Can be a mnemonic for symbol
      * Can be # followed by unicode value
    - Links: <a href = “relative path” target = “target”>hyperlink text</a>
      * Target \_blank for a new tab
    - Containers
      * Lists
        + Goes between <ul></ul> or <ol></ol>
        + Each item of the list goes in <li></li>
        + ul means bulleted list
        + ol means numbered list
      * Tables
        + Goes inside <table></table>
        + Content of a row goes between <tr></tr>
        + Content for each cell goes in <td></td>
        + Headers go into <th></th>
        + colspan = “cell #” in opening tag to increase cell width by a number of cells across
        + To add rows in Javascript, set a variable equal to tablename.insertRow();. Will add a row to the table, and set variable to represent row as object
        + To add cells in Javascript, set a variable equal to rowname.insertCell();. Will add a column to row, and set variable to represent the cell as object
      * Forms
        + Focus: while element is clicked
        + Blurred: when user clicks off element
        + Goes between <form></form>
        + To get values

Make the form a JavaScript object

Objectname.inputname.property

Possible properties: value (for text, radios), checked

* + - * + <input name = ”name” type = “” />

Possible types: text, checkbox, radio,

* + - * + Labels: <label for = “id” OR type = “type”>label</label>

Labels a form element

* + - * + Radios

Those of the same set share the same name

Each radio has a value that’s returned if the radio is submitted

* + - * + Textarea

A box to enter text, multiline, resizable

Strings placed between the tags are prefilled text in the box

Uses <textarea name = “name”></textarea> instead of the input family

* + - * + Submit is an input type that sends data to a server by running its onclick

JavaScript

* Can be put in a \*.js file
  + To use them in an HTML document, use src = “relative path”
  + Need another tag for local functions
* “ is interchangable with ‘
* End lines with a semicolon
* A function defines a task
  + To call (run) a function, parenthesis goes after its name
  + Inside the parentheses is the argument
  + Parameters are the arguments a function accepts
    - Creates temporary variables to pass values
  + To define a function:
    - function name(parameters)
    - {
      * code
    - }
  + It’s an object
  + Anonymous functions: no name, just used once
    - Ex: could be passed as an argument like this: function() {code}
  + callback function: a function passed as an argument for another function. The receiving function will call the passed in function at some point
* console.log(“string”) puts string in the JS console
* == compares 2 values after JS automatically converts them to a common data type
* === is a strict comparison that also requires both values to be of the same data type
* Variables
  + Name = value; to store
* Generating random number: use random function of the Math library
  + Generates random between 0-1
* parseInt(number) makes a number an integer
* (math symbol)= perform the math operation using right hand number onto left hand number
* document.getElementById(“id”) returns the object representing the element
* <HTMLElement>.innerHTML updates the content of an element
* && is and, || is or
* Use + to concat strings+= would concat strings and store it in string on left
* var creates temporary local variables, function scoped
* Let creates temporary local variables, block scoped
* To update CSS of an HTML object, objectname.style.(property) = value; (value is string)
* sessionStorage is an object that can store properties and values that persists after a reload
* Return makes a function equal a value. Script ends when return is reached
* Constants (const): variables that don’t change values
* While loops loop while a condition is met
* For loops: for (variable; condition; increment) { code }
* Events
  + Can be put as attribute of a HTML tag
  + The block of code to execute as a result of the event can use the variable event, which contains the Event object generated by the event. It contains all the details of the event
  + <Event>.preventDefault(); blocks the browser from executing its default response to the event
  + Onclick
  + Onmouseup, onmousedown, onmouseover, onmouseout, onkeypress
    - Onkeypress is only valid for elements in focus. An element has focus once it’s clicked on
  + Event listener: code that repeatedly checks if something has happened yet
    - The addEventListener method of HTMLElements can add event listener to the element
* Synchronous code runs line by line; the next line can’t run until the current line is done running
* Asynchronous code can execute at any time, regardless of what’s currently running

DevTools

* F12 or Ctrl + shift + i
* Sources tab shows the code
* Console tab shows JS console
* Breakpoint: pauses execution of JS at that line and allow access to all variable values that are in memory

Attributes

* Goes in body
* <body attribute = “value”>
* Event attributes take events as values
* onload is an event attribute that executes when the page loads

Control flow

* If (condition) { code to run if condition is true }
* switch(expression) {
* case value:
* Code
* break;
* case value:
* Code
* break;
* default:
* Code
* }
  + Break defines the end of the case’s code
  + Value must === expression for it’s case’s code to execute

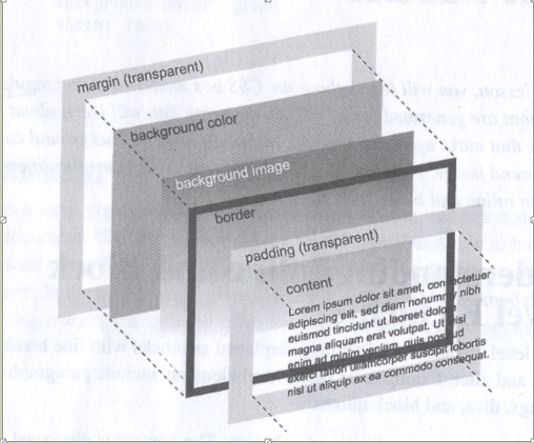
CSS

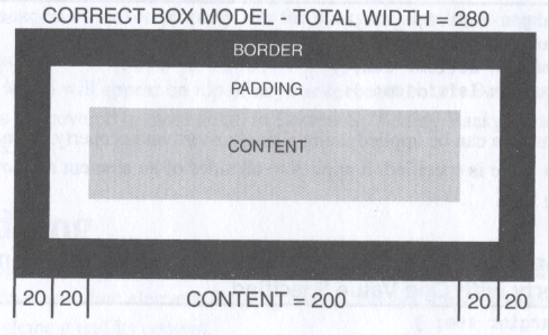
* Can be in a \*.css file
  + To use: <link rel = “stylesheet” href = “relative path” />
  + Linked stylesheets further down will take final influence over the rules defined in above stylesheets
* Comments go in /\* \*/
* Tag name to change
* {property: value;}
* Properties can be inherited from ancestors
* #name to apply CSS to a specific ID
* Color for text color
  + Can be by name, hex, rgb, rgba
* background-color for background color
* Background-image: url(‘’);
* Background-repeat
* Background-position
* Background-attachment
* Margins
  + They are relative to surrounding elements
* text-align for text alignment
  + Works for non inline contents
* width and height
  + In percentage of parent’s size
* float: everything wraps around the floating element
* padding-(direction): creates padding of defined pixels
* font-family to change font
  + Shorthand: font : weight size family
  + Weight can be a number from 100-900, bold, etc
  + Size can be in pt, vh (percentage of window height to occupy), vw (percentage of window width)
* text-decorations: underline, strikethrough, overline
  + Can also have styles and color with it
    - Shorthand: decoration style color
* @font-face rule can be used to get a custom font from online
* list-style-type to define which type of bullets to use in a list
* List-style-position defines location of bullets
* Table borders: border: width style color
  + Need to apply to table and td tags separately
* border-collapse to eliminate double borders in table
* writing-mode can make a text written sideways
* Display
* Opacity
  + Runs from 0 (invisible) to 1 (opaque)
* Selectors
  + Indicates relationships between HTML tags
  + Descendent: Space means to apply to only descendents of the previous tag
  + Comma: style applies to all elements
  + Plus: apply only to immediate sibling of the previous tag
  + Dot
  + Hash: by ID
  + Colon
* Class
  + A classification / group of elements to be styled and behave the same way
  + Elements are added to class through class attribute
  + Class styles are modified in style tag using dot selector. It means find elements of that class out of that type of tag. Or put nothing before the dot to apply to the whole class
* Pseudoclasses
  + Colon operator
  + Creates classes based on events happening to elements

Data types

* Strings
  + Are array-like objects
  + Use + to concat strings in JS
  + += would concat strings and store it in string on left, in JS
  + Can concat an HTML tag into a string, so that the element gets added to the body when innerHTML. Do this by putting the tags in “”
  + .length property gets number of characters in a string
  + .indexOf finds the first starting index of a character or string
    - If the argument isn’t found in the string, -1 is outputted
  + .substring: returns part of a string. First argument is starting index. Second is ending index. The ending index is excluded. If a second argument isn’t present, the method returns the rest of the string after the starting
* Arrays
  + Indexed list
  + Index count starts at 0
  + Arrays are objects, and are made of objects of any data type
  + List of data, all inside a [], separated by commas.
  + item from a specific index can be called up by arrayname[indexnum]
  + Calling index that doesn’t exist will create the index
  + After calling an index, its value can be changed by using = operator
  + Use array.push(item) to add to the end of the array
  + Use array.unshift(item) to add the start of the array
* Objects in memory
  + Objects, elements and variables are stored on the stack
  + For objects and elements, the value of them are stored on the heap, and on the stack, it only keeps track of the heap address of the value
  + Setting objects and elements equal makes both share the same heap address, so updating one of them updates the other
* string.splice(“string”); breaks a string at points where the argument string is. Resulting substrings are stored in an array
* Objects
  + Defined between {}
  + Has properties and methods, accessible and assignable by .
  + JSON: represents an object
  + {property:”value”, property:”value”, … property:”value”}
  + this: Allows an object to refer to itself
  + JSON.stringify converts an object into a string
  + JSON.parse(string) converts string to object
  + Classes
    - template for objects, built through a function called a constructor
    - By convention, the name of the constructor (class name) starts with a capital letter
    - Constructors are written like a standard function, but
    - this refers to the object the constructor is constructing, used to give properties to the new object
    - prototype: template for the parts of the object to be constructed
      * Is a property of the constructor
      * Add methods to an object via the prototype: Class.prototype.methodName = function () {code}
      * Inheritance: make the prototype of one class equal to that of another
  + To create an object/instance using a constructor, use new Constructorname();

Box Model





* Positioning
  + Static (default): elements are laid out in relation to previous elements in order
  + Relative: element position can be modified, relative of where it would be if it was static
  + fixed: like relative, but relative to the viewport
  + Absolute: like relative, but relative to the non-static parent element

Serverside - Node.js

* Node.js: an GUI-less distribution of Chromium designed for running server side code written in JS: processing requests and responding to them
* To run a server, open a terminal. Change its directory to the directory containing the server. Then, run node [file name of server]
* The terminal becomes the console
* Operates independently from client side code
* Node modules
  + Open sourced collection of libraries
  + require(module name) constructs and returns an object representing the node module module name
  + Node.js comes with some modules built in, such as fs. To get more modules, check out NPM
    - npm install <module name>
  + Package.json list what modules a project is using
  + Package-lock.json allows you to force which version of a node module is used
  + Fs
    - Stands for File System
    - Comes included with Node.JS, just import it
    - fs.readFileSync(path) synchronously returns a Buffer representing the file at path
    - fs.readFile(path, callback(err, data)) is the asynchronous version
      * Instead of the function returning information, the information is passed into the callback
      * The callback is not called until the main function is done gathering the information it should gather
      * The next line most likely would’ve run before the callback even gets executed
      * A typical asynchronous callback would be a function, with a parameter representing an error, and a parameter representing whatever information the original function should be getting. Ideally, you should check if an error has been passed into the callback. If so, execute code designed to handle the error. Otherwise, if no error, do what you gotta do.
  + To create your own package
    - Create a js file containing its code
    - For require to get the content, the variables and functions and stuff must be a property of exports
    - To import, the argument of require, assuming the js file is in the same directory as the server, is “./filenamewithoutjs”
* To open a server
  + Import the http module (comes included with node.js)
  + <http>.createServer(callback).listen(portnumber);
  + Callback has a parameter that takes in a request, and a parameter for an http response
  + In the callback, take the ServerResponse and writeHead(statuscode, {“Content-Type”: contenttype})
    - Status codes
      * 404: can’t locate requested content
      * 500: server error
      * 200: success
    - Content types
      * text/plain
      * text/html
      * text/js
      * text/css
      * image/png
  + In the callback, also take the ServerResponse and .write(content)
    - Text types are sent as strings
    - Everything else are sent as binary
  + ServerResponse.end() sends the response to the requester
  + Ctrl+c closes a server
  + Https requires you to have a certificate
  + Make requests to your server by visiting <your local ip address>:portnum/path
* AJAX
  + Asynchronous Javascript And XML
  + XHR: XML Http Request
  + To make a request, first create a new instance of XMLHttpRequest
  + Open the request: <XMLHttpRequest>.open(method, request, whether request should be async);
    - Methods
      * GET
      * POST
  + Then, detect when the server responds to the request
    - <XMLHttpRequest>.onreadystatechange = callback
    - In the callback, make sure that the request’s readyState property is 4 (response received), then run the code to process the response (the request’s responseText or response property)
  + Lastly, send the request: <XMLHttpRequest>.send();
  + The server would respond to the request the same way
    - Just add conditional cases to differentiate between handling different requests
  + Query string for sending info to server: <path>?property=value&property=value&property=value…