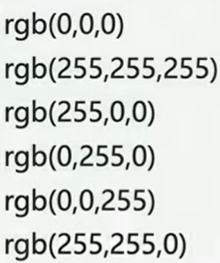
* <https://unbrokenbond.github.io/Html/>
* ! followed by [tab] will auto generate basic setup
* Hosting – need a server, domain name (unique – registered with DNS), FTP (file transfer to the server),
* Comments
  + HTML <!-- -->
  + CSS /\* \*/
* CSS Syntax 1. Body {background-color: lightblue;}
  + Body is the selector
  + Background-color is the property name
  + Lightblue is the value
  + 
* CSS Precedence & Inheritance
  + In line > imbedded > external
  + Try to only use external
  + 2 values set on same level 🡪 the second selector takes precedence
* CSS Fonts
  + Basic fonts everyone has or Google fonts – import or DL them
    - Basic fonts can just be typed into code without importing
      * Body {font-family: Arial;}
    - Imported fonts must be @import url(‘https://link’);
      * Body {font-family: ‘Rock Salt’, ‘Bradley Hand’, sans-serif
        + If there is a space, you must put it in ‘’
    - You can provide alternative fonts if your font is not available (prefer not to have a random font that is generated)
      * Body {font-family: Arial, Verdana, sans-serif;}
  + Font-size, letter-spacing, line-height,
* CSS Colors
  + Color names, RGB values, hexadecimal codes
  +  
  + Hexadecimal #
    - First 2 digits = red
    - Next 2 = green
    - Next 2 = blue
    - FF is highest value, 00 is lowest value
  + RGB
    - Can add an alpha or ‘a’ to the value
      * Transparent 0.0 – 1.0 solid
  + Test contrast contrast-ratio.com
* Week3
  + Class and ID Attributes
    - Element name (p, h1, etc.)
    - Class = any name you desire (no number)
      * Useful when only wanting to apply css to specific things and not all of the p’s, etc.
      * Placed in the beginning tag of the element <p class = “???”> </p>
      * .”???”{color: green;}
        + We put a “.” At the beginning to tell css to look for a class name
    - ID attribute = different from class as it is only applied to 1 element
      * In html <h1 id=”myheader”> html
      * In css #”myheader”{attributes;}
        + “#” at the beginning to tell css that it’s an ID
  + HTML File Basics
    - Naming rules:
      * Save file names in all lower case, only use alphabet, dashes or underscore
      * Index file will always be recognized as the home of your website by default
      * ! followed by tab 🡪 basic html setup
      * Meta tags – belong in the head section, they add additional info for your website (not shown on website)
        + Charset = UTF-8 (Unicode) 🡪 character set we are using
        + http-equiv = :”X-UA-Compatible” content = “IE=edge”
        + 🡪 tells Microsoft to use latest rendering edge
        + Name=”viewpoint” content=”width=device-width, initial-scale=1.0” 🡪 allows developer to take control of the visible part of the webpage

Width – screen size to device (phone, tablet, computer)

Scale – initial zoom when someone first comes to the page

* + - * Validate code – w3c = validator.w3.org 🡪 will validate your code
        + Tells you if you have any errors
        + Google ranks better if its validated
  + Wireframes
    - Depict what webpage will look like – present to client to see if there are changes needed before coding it.
    - Group 1 = Nav and logo, and footer stay the same from page to page
    - Group 2 = can be grouped smaller
      * Group 2a = banner image
      * Group 2b = product gallery
        + Can smaller group each product with a description
      * These are just examples of grouping
        + Helps with organizing our code and talking with clients
    - Deciding which element to use?
      * W3schools.com
        + Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.
        + Examples of **semantic** elements: <form>, <table>, and <article> - Clearly defines its content.
    - Placeholder images
      * Use <img src =’https://placeimg.com/size/size/animals alt=’placeholder img description’>
  + Troubleshooting
    - Validate HTML code at  <https://validator.w3.org/>.
    - Validate CSS code at <https://jigsaw.w3.org/css-validator/>.
    - Work from the first error 🡪 check code again to see if its fixed 🡪 repeat
  + Images
    - Stock image sites
    - Avoid blurry or pixilated – find larger pictures that you can make smaller if needed
    - Vector graphics – don’t pixilate, you can go big or small and it will keep its resolution
    - High resolution images can slow down the load speed – optimize image resolution – can resize and compress them. Remember to save your original image. 1000 pixels is the max width an image will ever be.
      * Can use picresize.com
      * Can use tinypng.com to compress
    - Check load speed of your website @ developers.google.com/speed/pagespeed/insights
  + Frontend Development
    - What the user sees
    - Technologies you’ll need to learn
      * HTML, CSS, and JavaScript
    - Tools
      * Text Editor – Visual Studio Code, Sublime Text 3, Atom
      * Browser Developer Tools – Chrome Dev Tools, Firefox dev tools
      * Version Control – Git & GitHub
    - Other Important Technologies
      * Sass / CSS Preprocessing
        + Adds useful features to css; organization
      * JavaScript Libraries: jQuery, React
        + Add complex javascript without having to code it
      * JS Frameworks: Angular, Vue.js
        + These are newer libraries for javascript, usually for lots of data or complex data
      * Front End Framework: Bootstrap
        + Provides variety of design templates; photo carousels, unique buttons, etc.
      * Command Line, CLI
      * Module Loading/Bundling Tools: webpack, Rollup, Browserify
        + Bundle code to help if run faster
      * WordPress
        + Content management system
      * Image Editor – Photoshop$, Sketch$
      * Wireframing Tools – Balsamiq$, Figma$
        + Useful when creating wireframes and collaborating with others on it
  + Backend Developer
    - * Work with frontend
      * Work on data (database) the website presents
        + User data: profiles, passwords, money, etc.
        + Store data in a way that ensures it is only presented to the person/people who are supposed to have access to it
        + Payment processing: accepting data, securing it, and making sure the charges go through correctly
        + APIs (application programming interfaces): not meant to be accessed by the web browser, but mobile phone apps and other programs

Written mostly in JSON or XML

* + - * Need to know how to do frontend development usually
    - Web server software – backend don’t usually code in this, you download, it collects requests from the browser. Determines based on web framework where to send the request.
      * Ex: Apache (small scale perfect for personal server), NGINX, Microsoft IIS,
    - Web framework
      * Libraries for programming languages that help the developer handle web requests
      * Ex: JavaScript (Node.js, Express), Java (Spring, JavaServer Faces)(used by amazon and google), Python (Django, Flask)(great for data analysis), C# aka Microsoft (ASP.NET MVC, ASP.NET Core)
    - ORM (Object-Relational Mapping)
      * Connect to a database
      * Converts information in the database to objects for the server
      * Writes SQL
        + Retrieves information from the database (query) if ORM cannot find and retrieve it
    - Database Software
      * MySQL, SQL Server, PostgreSQL
    - Version control software
      * Git and Subversion
    - Cloud Hosting Platforms
      * Amazon Web Services, Heroku, Google Cloud Plateform

Browser requests info from server 🡪 Web Server Software works with Web Framework to handle the request from the database 🡪 code written in Web Framework uses ORM to request information from the database

retrieve from Database 🡪