FRONT-END VOCABULARY

UNIT TWO: HTML/CSS

HTML

- **HTML**: Hypertext Markup Language - the language for defining the content and structure of a webpage.
- **HTML5**: The latest version of HTML. It has been the standard for many years now.
- **CSS**: Cascading Style Sheets - the language for styling HTML
- **Developer (Dev) Tools**: A panel built into the browser with a lot of features for investigating, debugging, and experimenting with HTML, CSS, and JavaScript on a webpage. One way to open it in Chrome is to right-click and select "Inspect".
- **HTML Element**: a piece that makes up the HTML document. There are different kinds of elements for different things (e.g. headings, paragraphs, buttons). Elements are usually defined by a start tag, some content, then an end tag. But empty (or void) elements have no content, and are defined by just a single tag.
- **HTML Tag**: used in HTML to define an element. They include the name of the element surrounded by angle brackets < >. Most elements need two tags, a start and an end.
- **HTML Attribute**: additional information supplied in the start tag of an element. Attributes usually consist of a name and a value. The value is usually surrounded by quotes ("").
- **Nesting**: HTML elements can be placed inside of each other. The inside (child) element's tags are placed between the start and end tags of the outside (parent/container) element.
- **Comment**: a portion of code that provides some explanation or clarity for humans reading the code but is essentially ignored by the computer.
- **HTML Form**: an element used as a container for various form input elements. It includes behavior for sending user input data to the web server, but it can also be used with JavaScript to collect the user input data.
- **Semantic Elements**: These HTML elements have names that describe their purpose. They provide great value for both readability and *accessibility* compared to using generic elements such as and for everything. Examples include <footer>, <nav>, <article>, and <table>.
- **Accessibility**: ensuring that a product or service is usable by people with a range of abilities. For example, consider how people with visual impairments or color blindness or who only use a keyboard to navigate will experience your website.

CSS

- **CSS Rule**: a CSS selector combined with one or more declarations that identifies and defines styles for a set of HTML elements.

- **CSS Selector**: the part of a CSS rule that specifies which HTML elements to target. Types of selectors include element, id, class, attribute, descendant, and more.
- **CSS Declaration**: a property/value pair in a CSS rule that specifies a single style feature for the targeted elements. Multiple declarations must be separated by a semicolon (;).
- **CSS Property**: the name of a style feature. (e.g. margin, color, font-size)
- **CSS Units**: Some CSS values require units for specifying length/size. (e.g. px, rem, %, vh)
- **Pseudo-class**: used in a selector to target specific states of an element. Pseudo-classes begin with a colon. (e.g. :hover, :visited, :first-child, :nth-child, :nth-of-type)
- **Pseudo-elements**: used in a selector to target specific parts of an element. Pseudo-elements begin with two colons. (e.g. ::after, ::first-line, ::selection)
- **Specificity**: how the browser determines which CSS rule wins when there is a conflict on a given declaration.
- **Box Model**: the four adjustable parts that make up the rectangle layout of an element. From inside to outside, the parts are content, padding, border, margin.
- **Display Inline**: value of the display property that causes an element to flow in the direction of text. For English content, this is left-to-right, wrapping at each line.
- **Display Block**: value of the display property that causes an element to stack vertically, generally taking the entire width of the container and falling below the previous element and above the next element.
- **CSS Positioning**: a way of modifying or overriding the position of an element in the document using a combination of the position, left, right, top, and bottom properties.
- **CSS Transition**: animations of CSS properties between different states of an element.
- **Vendor Prefixes**: For some newer CSS properties, different browsers may require a prefix to the property name. (e.g. -moz-, -webkit-, -ms-)
- **Responsive**: designing and developing a website to work well on a continuum of sizes and types of devices. This includes factors such as screen size, screen orientation (landscape vs portrait), touch screen, and others.
- **Mobile-first**: responsive design and development that starts with the mobile experience (e.g. smaller screens) and then expands to the tablet, desktop and other larger device experiences.
- **Flexbox layout**: a configurable CSS layout where child elements are laid out along one dimension within their parent. This includes direction, justification, alignment, and wrapping.
- **Flexbox axis**: Flexbox includes two configurable axes--the main axis (where justify-content applies) and the cross axis (where align-items is used).
- **Grid layout**: a configurable two-dimensional CSS layout.
- **Media queries**: allows some CSS rules to take effect only based on specific attributes of devices. Commonly min-width and max-width are used to enable certain rules at specific screen width ranges. The exact widths specified are called *breakpoints*.
- **Viewport Meta Tag**: an HTML tag in the <head> that tells the browser how to adjust the webpage for mobile devices. We always specify this because the default behavior is

to zoom rather than giving CSS the control to use responsive development and media queries effectively.

UI/UX

- **UI**: User Interface - usually refers to the graphical interface (visuals, colors, and layouts). It is how things look. UI makes interfaces *beautiful*.
- **UX**: User Experience - relates to creating products that will provide personal and meaningful experiences. It is how things work. UX makes interfaces *useful*.
- **Bootstrap**: a front-end toolkit for quickly creating mobile-first responsive websites. It provides out-of-the box CSS and components and a layout system. Bootstrap is just one of many such toolkits available.
- **User Persona**: a short bio of a fictional person that represents a type of user. It helps you "get into their shoes" as you design your product.
- **Prototype**: a quick-to-build sample of a product that test users can give feedback on before a design is built out in its full form. This can be anything from a paper sketch to a digital version built with a specialized program.
- **Wireframe**: a skeleton of what we want a site to look like; the blueprint or sketch of the layout the site should include. Wireframes do not include visual details such as exact sizes, shapes, colors, and images.
- **Pair Programming**: a technique where two programmers work together at one computer. One takes the role of the "driver", operating the computer and typing the code. The other takes the role of the "navigator", helping to proofread and discuss strategy and decisions.

Version Control / Git

- **Command Line / Command Prompt / Terminal**: a program for running programs and giving the computer instructions by typing them. It is the best or only way to do many development tasks, and it allows certain tasks to be automated.
- **File Path**: a series of folder/file names, separated by slashes, that indicates the location of a file.
- **Absolute Path**: the full path to a folder starting at the root. The absolute path always starts with a slash or the drive letter such as “C”. (e.g. /Users/grant/projects/lab1)
- **Relative Path**: gives the location of a file or folder relative to the current directory (e.g. images/cow.png)
- **Root Directory**: The "highest" folder. Every other folder is somewhere inside this. On Windows, it's usually "C:\". On Mac it's a single forward slash "/".
- **Home Directory**: Each user of the computer has their own home directory that’s named after their username. This is the folder that has all their stuff such as documents, pictures, and desktop in it. (e.g. /Users/grant) The shortcut for this folder on the command line is tilde (~).

- **Version control**: (sometimes called source control) a system that manages history and changes to a program, website, or other collection of files.
- **Git**: an open source, distributed version control system originally developed by Linus Torvalds.
- **GitHub**: a social coding service that offers hosting for software projects that use Git as their source control.
- **Git Repository (Repo)**: a database where Git stores snapshots of all the files in a project as they change over time. If you have multiple projects, each will need its own Git repository.
- **Local Repository:** a repository on your computer where Git tracks files in a folder.
- **Remote Repository**: a copy of a Git repository stored online, for example on GitHub.com.
- **Working Directory**: the folder where project files are actively worked on.
- **Commit**: a snapshot of the entire project at a point in time. e.g. git commit -m 'did a thing'
- **Staging Area**: where changes are queued up to be committed. Changes must be added to the staging area before a commit. e.g. git add readme.md
- **.gitignore**: a file that lists files that should not be committed to Git. The file name starts with a period to indicate that it is a hidden file.
- **Git Status**: a command that displays information about the state of files and commits in the current Git repository. (e.g. git status)
- **Initialize (Git Init)**: Turn the current folder into a brand new Git repository. (e.g. git init)
- **Clone**: Copy a remote repository (often from GitHub) to your computer. It is copied into a new folder.
- **Branch**: There can be parallel versions of the files in a Git repository. The versions are called "branches" and each has its own name. Branches are useful to support an experimental version, an old version, or to have multiple developers working on the project without interference.
- **Master Branch**: every Git repository typically starts with one branch called "master". This tends to be the main version of the code.
- **Merge**: The process of bringing changes from another branch into the current branch. (e.g. git merge color-changes)

UNIT THREE: JAVASCRIPT

- **JavaScript**: A programming language that allows us to give step-by-step instructions for how a webpage or other program should behave, for example when a user clicks a button or fills out a form.

- **Dynamically-Typed**: JavaScript is a dynamically-typed language. Variables don't have pre-set types. When the program runs, any kind of data can be used with each variable. In other statically-typed languages, variable types are pre-set or fixed in the code.
- **Object-Oriented**: JavaScript is an object-oriented language. This means that data and code can be bundled together into packages called objects. In a JavaScript, the data is called fields or properties (like variables that belong to the object), and the code is called methods (functions that belong to the object).
- **Variable**: a named container for values (data). It allows us to store data in computer memory and use it again later.
- **Declaration**: The code to create a variable. Use one of three keywords: let, const, var.
- **Assignment**: set the value of a variable.
- **Initialization**: the first assignment for a variable.
- **Constant**: A variable declared with *const*. The value can be set once, then cannot be re-assigned.
- **Block Scope**: A variable declared with *let* or *const* has block scope. It can only be used within the code block where it is declared.
- **Code Block**: a portion of code within a set of curly braces ({ }).
- **Hoisting**: a function declared with the *function* keyword or a variable declared with the *var* keyword are hoisted, meaning they can be used above or below where it is declared in the file. Variables declared with *let* or *const* are not hoisted and can only be used below where they are declared.
- **Data Type**: The various types of values that can exist in JavaScript. e.g. strings, numbers, objects, booleans, etc. Every piece of data in JavaScript has a specific type.
- **Character**: a single letter, number, punctuation mark, space, emoji, or other symbol.
- **String**: a sequence of characters. e.g. "Hello" or "As easy as 123!"
- **JavaScript Object**: collection of key/value pairs; a way to package multiple values together into one larger unit.
- **Key**: a label for a value in an object--used to access (e.g. get and set) an individual value.
- Array:
- **Undefined**: One kind of empty value--when a value does not exist or has not been set.
- **Null**: Another kind of empty value--the intentional absence of a value.
- **Concatenation**: + operator with strings. Makes a new string by joining the strings together.
- **Template Literal (Template String)**: surrounded by backticks (`). Creates a string that includes values from JavaScript expressions. e.g. let greeting = `Hello ${name}!`;
- **Truthy**: a value that is considered true in a boolean context. [source MDN] (e.g. true, non-zero numbers, non-empty strings)
- **Falsy**: a value that is considered false in a boolean context. [source MDN] (e.g. false, zero, empty strings, null, undefined, NaN)
- **Conditionals**: if/else statements, switch statements, and the ternary operator.

- **For Loop**: repeatedly executes a block of code using a counter variable. Useful for looping a fixed number of times.
- **While Loop**: iterates as long as a given condition evaluates to true. The condition evaluates before each iteration of the loop, so it may not execute at all if the condition starts out false.
- **Do... While Loop**: iterates as long as a given condition evaluates to true. The condition evaluates after each iteration of the loop, so it will always run at least once.
- **For... In Loop**: iterates over keys of an object (including array indexes).
- **For... Of Loop**: iIterates over values of an object (including array values).
- **Break**: Immediately exits a loop or switch statement.
- **Continue**: Skips to the next iteration of a loop.
- **Function**: a defined block of code for a particular task. It does not necessarily run immediately. It can be called (invoked) when needed.
- **Parameters**: the inputs to a function. Some functions require some input values to do their task. Parameters allow these values to be provided when calling the function.
- **Arguments**: the values that are given to a function for its parameters when it is called.
- **Return statement**: 1) immediately exits the function. 2) may specify a value to send back to where it was called.
- **Function Declaration**: Function keyword followed by function name. Only this type of function is hoisted.
- **Function Expression**: Rather than naming the function, assign it to a variable in order to give it a name.
- **Arrow Function Expression**: Uses "=>" rather than the "function" keyword and has several shorter syntax variations. Arrow Functions also preserve the value of the "this" keyword.
- **Anonymous Function**: A function expression or arrow function that is never assigned a name. This is common for callbacks.
- **Closure**: a function that has references to variables and functions that are coming from an outer scope.
- **Method**: a function that belongs to and object or class.
- **Class**: a blueprint cookie-cutter for objects. Individual objects are instantiated (created) from the blueprint or cookie-cutter.
- **Instance**: an object made from a class.
- **Constructor**: the function for each class that builds object instances from that class.
- **Prototype**: holds all the common properties and methods for all instances of a class.
- **this**: a keyword that, when used inside methods, generally references the current object.
- **Array**: an ordered collection of data with numbered indexes.
- **Rest operator**: collects all the remaining arguments of a function into a single array
- **Spread operator**: used to copy elements from one array or object into a new array or object literal

- **Destructuring**: a shorthand for extracting values from objects or arrays into individual variables.
- **Debugging**: the process of identifying and fixing errors
- **Exception**: an error in the code as it runs that stops the code at that point. This will generally display an error message in the console, or it can be handled with a try-catch statement.
- **DOM**: Document Object Model - an interface that allows JavaScript to access and modify HTML
- **Event**: something that happens on a webpage, such as: the page loads, the user clicks, scrolls, moves the mouse, or types.
- **Event Target**: the element on the page where the event happened, for example the button element that was clicked.
- **Event Listener (Event Handler)**: a function that is registered to be called when a particular event happens on a particular element.
- **Event Bubbling**: Many events run handlers up the HTML element tree. For example, a click event on a button also triggers click handlers on the <p> that the button is in, the above that and the <body> that contains everything. Events bubble to "ancestor" nodes (i.e. container elements) in the HTML tree.
- **Event Delegation**: registering a handler function on an ancestor node (container element) to handle events on one or more of the descendant elements. For example, add a listener to a table in order to handle clicks on any of the cells in that table.

UNIT FOUR: ADVANCED TOPICS

Data Structures & Algorithms

- **Data Structure**: a way to store and organize data that makes it efficient to access and manipulate
- **Abstract Data Type**: a type defined by what values and behaviors it has. It defines what you can do with data. Unlike a data structure, it does not define exactly how the values and behaviors are stored in memory. There are often multiple different data structures that could be used to implement an abstract data type, each with pros and cons.
- **List**: a numbered collection of values in a specific order. Items can be added and removed.
- **Set**: an abstract data type with no duplicates, and often no specific order. Items are either in or out.
- **Map**: an abstract data type where values are stored with keys (a.k.a. names). Great for looking things up quickly.
- **Queue**: a list where items enter one end and exit the other. Also referred to as FIFO (first in - first out). It handles them in the order they came in.
- **Stack**: a list where only one end is accessible. Also referred to as LIFO (last in - first out).

- **Tree**: an abstract data type where nodes are connected in a tree structure, starting at the root and branching out.
- **Graph**: an abstract data type where nodes can be connected in many ways.
- **Array List**: a List implemented using an array data structure. Since the array data structure (unlike the JavaScript Array type) cannot grow or shrink, it also stores the current length.
- **Linked List**: a List implemented using a chain of Nodes. Each node stores a value and a pointer to the next node in the chain.
- **Hash Function**: any function that can be used to map data of arbitrary size to data of a fixed size. For a hash map the hash function returns an integer.
- **Algorithm**: a finite set of steps that expresses how a specific task should be done.
- **Flowchart**: a specific diagramming technique for laying out the steps of an algorithm.
- **Pseudocode**: the specific steps of an algorithm written in text for a human to read. Similar to computer code, but not language-specific and generally requires less focus on syntax.
- **Greedy Algorithm**: a type of algorithm that doesn't look ahead and only looks for obvious and immediate benefit.
- **Divide and Conquer Algorithm**: a type of algorithm that splits larger problems into smaller problems repeatedly until the problems are small enough.
- **Search Algorithm**: an algorithm that finds something in a list. e.g. linear search, binary search
- **Sort Algorithm**: an algorithm that takes a list and puts all the elements in order. e.g. insertion sort, selection sort, merge sort
- **Complexity Analysis**: the measure of the efficiency of an algorithm
- **Big-O Notation**: a notation for describing the worst-case efficiency of an algorithm using a mathematical expression. e.g. O(1), O(n), O(n log n)

Testing & TDD

- **Software Testing**: the process of ensuring that the code works the way the developers intended it to without bugs.
- **Manual Testing**: humans use the application, usually trying every single feature to make sure it works correctly.
- **Automated Testing**: computer programs are written to execute various features of the application to make sure it works correctly.
- **TDD**: Test Driven Development - the process of writing code one test-case at a time by writing an automated test first and then the code for that case. These small steps follow the red-green-refactor cycle and tend to produce maintainable code.
- **Refactoring**: making changes to code to improve the solution without changing its behavior. Once code is working, there are often opportunities to make it more clean, clear, and concise.

- **Test Framework**: provides syntax for writing tests and checks the results. e.g. Jasmine, Karma, Jest
- **Arrange, Act, Assert**: a common pattern for writing a test case. *Arrange*: Get a few things in order before tests can run--values, systems, etc. *Act*: Run the code. *Assert*: Verify that the code did what you expected.
- **Spies**: used in testing, spies are fake functions with special abilities to track how they are called.
- **Mocks**: fake versions of real modules, services, and other dependencies used for testing.

UNIT FIVE: ANGULAR

- **TypeScript**: a strongly typed, object-oriented, compiled language. It compiles to JavaScript.
- **Strongly-Typed**: Variables and functions have pre-set, unchanging types in the code.
- **Compile**: Convert source code into a lower-level form such as machine code. TypeScript can't run directly in the browser or Node.js, but JavaScript can. So a process converts the TypeScript source code into equivalent JavaScript code.
- **Interface (in TypeScript)**: defines properties and methods that objects must have--or in the case of optional properties--may have.
- **export**: a keyword in TypeScript/JavaScript that makes a value or function available to other files.
- **import**: a keyword in TypeScript/JavaScript that brings in values and functions from another package or from another file within the project.
- **Angular**: a platform and framework used to build client applications in HTML and TypeScript.
- **Component Based Architecture**: Components are small pieces of a user interface that are independent of each other. An application is constructed by creating these components and combining them into a whole.
- **Single Page Application (SPA)**: an application where much of its functionality can be performed without leaving or refreshing the webpage. This tends to lead to a smoother user experience.
- **Angular Module (NgModule)**: A unit in angular for grouping related components, services, and other things with a common purpose. A large Angular application can be made of many modules, but our applications generally focus on just one "root" module. (Technically, routing in our apps is set up in its own module.)
- **Angular Component**: Components are the "views" or pieces that make up the user interface. Each component has a TypeScript class, an HTML template file, and a CSS file.
- **Angular Service**: an element of an Angular application that is used for data and logic that isn't associated with one specific view and typically will be shared with multiple components. We often use these to hold data and communicate with APIs.

- **Angular CLI**: Command Line Interface - a command line tool that is used to create and run Angular applications and to generate elements of an application such as components and services.
- **Template**: the HTML portion of a component that defines what the user sees and can interact with. It is mostly HTML, but includes extra Angular syntax.
- **Data Binding**: The template defines how the HTML will be presented based on the data in the component's TypeScript class. Data binding is the automatic process of synchronizing the data in the component with the template. It can be one-way or two-way.
- **Template Expression**: used with interpolation and attribute/property binding. Allows a subset of JavaScript plus pipes and several additional operators.
- **Template Statement**: used to respond to an event. Typically calls a method on the component class.
- **Interpolation**: used in templates to inject text into the HTML. The syntax is double curly braces ({{ }}).
- **Attribute Binding/Property Binding**: uses a template expression to specify values to attributes or directives. The syntax is square brackets ([ ]).
- **Event Binding**: specifies template statement to run when an event occurs. The syntax is parentheses (( )).
- **Two-Way Data Binding**: Used with ng-model and a form input, two-way data binding synchronizes the data between the class and the form input. On one hand, if the data changes the input value changes. On the other hand, if the user types into the input, the data changes. The syntax uses the both square bracket and parentheses ([( )]).
- **Structural Directive**: change the DOM layout by adding and removing DOM elements. e.g. ngIf, ngForOf
- **Attribute Directive**: change the appearance or behavior of an element, component, or another directive. e.g. ngClass, ngStyle
- **Pipe**: takes in data as an input and transforms it to a desired output. Used in a template expression with the pipe/vertical bar character (|). e.g. | date, | lowercase
- **Input Property (Input Binding)**: a property is decorated with @Input(). This allows a parent component to set this property.
- **Output Property (Output Binding)**: a property is decorated with @Output(). This allows a parent component to respond to events emitted on this property.
- **EventEmitter**: type used for output properties, allowing a child component to emit events to a parent component.
- **Dependency Injection**: Other services and Angular elements can be used by adding them as parameters to the constructor. Angular automatically detects and sets the values for these parameters. This is called dependency injection.
- **Angular Router**: allows navigation from one view to another by interpreting the browser URL.

- **Route**: A particular browser URL path for the router to respond to. It will typically display a specified component or redirect to another URL path.
- **RouterOutlet**: a directive from the router library that is placed in the template where the router should display the component for the current route.
- **RouterLink**: an attribute directive that is applied to anchor tags. It tells the router which route to go to when the anchor tag is clicked.
- **API**: Application Programming Interface. As a general definition, an API defines the correct way for a developer to write a program that interfaces with a particular system. It is an interface where two systems meet and communicate. In this bootcamp, when we refer to APIs, we're usually referring specifically to Web APIs.
- **Web API / Web Service**: a remote server that our code can communicate with, usually to get data.
- **API Key**: a unique string provided when registering for an API. It is typically sent along with API requests and tracks limits and potentially costs of using the API.
- **OAuth**: a standard system for authentication (login) that is widely used for APIs.
- **OAuth Access Token**: a temporary unique string that is provided by an OAuth API after authentication is complete. It should be used with subsequent requests in order to prove that authentication has completed successfully.
- **JSON**: JavaScript Object Notation - a textual data format that is easily read by computers and humans. It uses JavaScript syntax for object literals, array literals, strings, numbers, and booleans.
- **AJAX**: Asynchronous JavaScript And XML: the technology within the browser that allows for API requests through JavaScript
- **Asynchronous**: Code keeps running while another process happens. It does not stop and wait.
- **HttpClient**: Built-in Angular service for making API calls.
- **Observable**: an object that represents potentially future events. We can subscribe to it to be notified if and when those events occur. HttpClient provides Observables that represent the future event of an API call completing (or failing) and perhaps providing some data.

UNIT SIX: NODE.JS

Node.js

- **Node.js**: a platform that allows JavaScript to be run outside of the browser.
- **node command**: run a JavaScript file from the command line
- **nodemon**: like the node command, but automatically reruns the file whenever an update is made to the project files.
- **npm**: Node Package Manager - used most often to download and install node packages (a.k.a. modules or libraries) that the project needs.

- **npm init** : command to set up a new project using Node.js. It creates a package.json file for the project.
- **package.json**: a file in the root directory of a Node.js project. It describes the project and lists the other packages that it depends on (known as dependencies).
- **npm install** : command to download and install a package in the project
- **npm install -g** : command to install a node package globally. The package becomes a command that can be run from the command line anywhere.
- **require**: a function in Node.js that imports from another package or from a file within the project.
- **module.exports**: in Node.js this is used to set the value or values that are exported from a file (i.e. a module). These are what is available to other files when they use *require*.

APIs & Express

- **API**: Application Programming Interface. As a general definition, an API defines the correct way for a developer to write a program that interfaces with a particular system. It is an interface where two systems meet and communicate. In this bootcamp, when we refer to APIs, we're usually referring specifically to Web APIs.
- **Web API / Web Service**: a remote server that our code can communicate with, usually to get data.
- **JSON**: JavaScript Object Notation - a textual data format that is easily read by computers and humans. It uses JavaScript syntax for object literals, array literals, strings, numbers, and booleans.
- **REST**: REpresentational State Transfer - an architecture style for designing APIs.
- **API Client**: the program that uses the API.
- **API Server**: the program that provides the API.
- **HTTP Request**: the message that the client sends to the server. This begins the exchange.
    - **HTTP Method**: part of a request that determines what to do (the verb). e.g. GET, POST, PUT, DELETE
    - **URI**: Uniform Resource Identifier - a string that identifies a resource. A URL is a type of URI. In Express, the *path* portion of the URL is also referred to as the URI.
    - **URL Path / URI Path**: part of the URL that determines which resource (the subject).
    - **Path Parameters / Route Params**: a variable portion of the URI path. Anything can be in that part of the URL and it will still run this route.
    - **Query String Parameters**: an optional portion of the URL that includes key-value pairs. It begins with a ?, and parameters are separated by &.
    - **Request Body**: POST and PUT requests include a body which holds the content (data) that needs to be sent to the server.

- **HTTP Response**:the message that the server sends back to the client. This ends the exchange.
    - **Status Code**: a numeric code indicating the outcome of a request, including whether it succeeded or failed. (e.g. 200 OK, 404 Not Found)
    - **Response Body**: the content (data) that needs to be sent back to the client.
- **CRUD**: Create-Read-Update-Delete - the four common operations to perform on data.
- **Postman**: an HTTP client that is useful for testing or investigating APIs.
- **Express**: a framework for Node.js that simplifies creating HTTP servers and REST API servers.
- **endpoint / route**: a specific URI (or URI path) of an API

UNIT SEVEN: DATABASES & FINAL PROJECT PREPARATIONS

- **Database**: an organized collection of data.
- **Relational Database / SQL Database**: a type of database that organizes data into one or more tables (or "relations") of rows and columns, usually with a unique key for each row. Data is accessed and manipulated by the SQL language.
- **SQL**: Structured Query Language - a programming language designed to manage the data held in relational databases
- **NoSQL**: Not Only SQL - a catch-all term for databases that operate outside the relational standard.
- **DBMS**: DataBase Management System - software that runs the database and allows other programs, such as our own applications or administration tools (PGAdmin)
- **PostgreSQL (Postgres)**: an open source Database Management System.
- **PGAdmin**: An administrative tool that allows developers to access the database visually and manipulate the data.
- **Table**: holds a particular type of data in columns and rows
- **Row**: rows in a table represent instances of that type
- **Column**: the columns represent various values attributed to that each row/instance
- **node-postgres (Node PG)**: a Node.js module for interfacing with your PostgreSQL database.
- **Connection String / Connection URI**: a URI that combines the various pieces of information needed to connect to a database, including type of database, host and port, database or schema name, and sometimes username and password.

UNIT EIGHT: FINAL PROJECTS

- **MVP**: Minimum Viable Product - an early version of software that is just barely good enough to provide the intended value to customers. The goal is to get something meaningful out in front of customers so that they can start using it and you can start

learning from their experiences to know what and how to improve. At Grand Circus, your MVP must deliver your core value proposition and meet all the minimum technical requirements for the final project.

- **Deployment**: the activities that make software available for use. This might include compiling, building, and optimizing. It will also include installing the final product to a place where it can be accessed by the intended audience.
- **Hosting**: installing your finished product on a server so that it is available for use beyond your local environment.