NODE NOTES

SYMBOLS

/ / - fences (serve as quotes for **REGULAR EXPRESSIONS).**

Release 1.0.1 - **1st #** is the major release; **2nd/middle #** is the minor release; **3rd/last #** is bug fixes.

CTL/CMD + C (console) - cancel/stop running script.

KEYWORDS

fs: (NODE.js) module provides an API for interacting with the file system.

join - (JS) The array.join() method is an inbuilt function in JavaScript which is used to join the elements of an array into a string.The elements of the string will be separated by a specified separator and its default value is comma(,).

module.exports - (JS) The module.exports or exports is a special object which is included in every JS file in the Node.js application by default. module is a variable that represents current module and exports is an object that will be exposed as a module. So, whatever you assign to module.exports or exports, will be exposed as a module.

NPM INIT (-Y) - (NODE.js) creates a single file package.json containing project meta information. The -Y skips the questions, answering yes to all.

parseInt() - (JS) translates input string into an integer.

parseFloat - (JS) This function determines if the first character in the specified string is a number. If it is, it parses the string until it reaches the end of the number, and returns the number as a number, not as a string.

process.argv - The process object is a global that provides information about, and control over, the current Node.js process. As a global, it is always available to Node.js applications without using require().

The process.argv property returns an array containing the command line arguments passed when the Node.js process was launched. The first element will be process.execPath. See process.argv0 if access to the original value of argv[0] is needed. The second element will be the path to the JavaScript file being executed. The remaining elements will be any additional command line arguments.

require - (NODE.js) function to export a main object. When Node invokes that require() function with a local file path as the function's only argument, Node goes through the following sequence of steps:
Resolving: To find the absolute path of the file.
Loading: To determine the type of the file content.
Wrapping: To give the file its private scope. This is what makes both the require and module objects local to every file we require.
Evaluating: This is what the VM eventually does with the loaded code.
Caching: So that when we require this file again, we don't go over all the steps another time.

return - (NODE.js) renders output to console.

switch (JS) - rather than else if, *switch* tests the variable, as to not repeat comparisons. It simply matches properties to cases.

GOOD PRACTICES

1. Always read the documentation for a package!

2. Install packages globally to make available on your whole drive.

3. MySQL is a great choice for Node.js

4. Use NODEMON during development to automatically restart local server!

VOCABULARY

FUNCTION EXPRESSIONS: A Function Expression defines a function as a part of a larger expression syntax (typically a variable assignment). Functions defined via Functions Expressions can be named or anonymous. Function Expressions must not start with "function" (hence the parentheses around the self invoking example below) e.g.

```
 //anonymous function expression
 var a = function() {
     return 3;
   }

  //named function expression
  var a = function bar() {
     return 3;
    }

   //self invoking function expression
   (function sayHello() {
       alert("hello!");
   })();
```

HTTP: Hyper Text Transfer Protocol

NODE.js (runtime environment) allows you to execute *JAVASCRIPT* on servers. Implemented in 2009, current industry standard.

NODE MODULE: Almost all npm packages (at least, those that are Node programs) contain many modules within them (because every file they load with require() is a module). In the context of a Node program, the module is also the thing that was loaded from a file.

PROCESS OBJECT: The process object is a global that provides information about, and control over, the current Node.js process. As a global, it is always available to Node.js applications without using require(). The process object is an instance of EventEmitter.

CONCEPTS
axios - (server side AJAX) a promise-based HTTP client that works both in the browser and in a node.js environment. It basically provides a single API for dealing with XMLHttpRequest s and node's http interface. Besides that, it wraps the requests using a polyfill for ES6 new's promise syntax.
fs: (NODE.js)The fs module provides an API for interacting with the file system in a manner closely modeled around standard POSIX functions. To use this module: const fs = require('fs'); All file system operations have synchronous and asynchronous forms. The asynchronous form always takes a completion callback as its last argument. The arguments passed to the completion callback depend on the method, but the first argument is always reserved for an exception. If the operation was completed successfully, then the first argument will be null or undefined.
MODULE PATTERN (COMMON JS MODULE SYSTEM/PATTERN) - The Module Pattern is one of the most common design patterns used in JavaScript and for good reason. The module pattern is easy to use and creates encapsulation of our code. Modules are commonly used as singleton style objects where only one instance exists.
package.json (Node.js) - lists all the required dependencies and provides version control**.**
package-lock.json (Node.js) - keeps a manifest of every dependency and helps optimize the package and cacheing. **!!!SHOULD NOT ALTER!!!**
RESTFUL APIs - can return XML, JSON, YAML or any other format depending on what the client requests.
NODE CALLBACK FUNCTIONS - written as error 1st. *(err, resp).*
SEMANTIC NUMBERING: Release **1.0.1 - 1st #** is the major release; **2nd/middle #** is the minor release; **3rd/last #** is bug fixes.

PROCESS VARIABLE: You can access variables on the process object in any file in your project because it is global. If this object was not global, the console object would not be accessible from any file either, it is really an object that refers back to the process object. The process object is global because it provides information about the current running Node process and therefore should be available from any file without having to require it.

REGULAR EXPRESSION: matches by a pattern, not a string.

BUILDING NODE PACKAGES:

1. NPM init - Creates a package.json

2. NPM install - Imports modules listed as dependencies in package.json.

3. NPM build

4. NODE index.js - Runs the Javascript file using Node.js

USE, GET, LISTEN *(EXPRESS):*

EXPRESS ROUTER CLASSES to create modular, mountable route handlers. A Router instance is a complete middleware and routing system; for this reason, they are often referred to as a “mini-app”.

The *app.listen()* method returns an **http.Server** object and (for HTTP) is a **convenience method.**