**REVISION ON NODE MODULES:**

**Modules**:

* are essentially reusable pieces of code that encapsulate specific functionality.
* play a crucial role in organizing and modularizing code.

**HOW?** -> They contribute to code

* maintainability
* reusability and
* separation of concerns.

**Types: -**

* Core modules
* Local modules
* Third-party modules

1. **Local modules:**

* are custom modules created by developers within a Node.js application.
* Used to encapsulate and abstract specific functionality that can be reused across different parts of the application.
* Are typically stored in separate files with a .js extension and can be organized into directories for better code management.
* To use, you need to export the desired functionality using module. exports and then require it in other parts of your codebase.

**In utils.js:**

// Exporting a function

exports. myFunction = () => {

// Functionality goes here

};

**In another file:**

const myModule = require('./utils');

myModule.myFunction(); // Call the exported function

console.log(myModule.myVariable); // Access the exported variable

1. **Third-party modules:**

* Are created by developers outside of the Node.js core team.
* Are published on package registries like npm.
* Provide additional functionality that extends the capabilities of Node.js.
* Cover a wide range of use cases, including web frameworks (Express.js), authentication libraries (Passport.js), database connectors (Mongoose), and many more.
* To use, you first need to install it using ***npm or yarn, and then require it in your code.***

Here's an example of installing and using a third-party module:

$ npm install express

In `index.js:

``javascript

const express = require('express');

const app = express ();

// Express application setup

// ...

app.listen(3000, () => {

console.log('Server started on port 3000');

});

1. **Core modules:**

* Set of built-in modules that provide essential functionality.
* Core modules can be readily accessed without the need for additional installation or configuration.
* Covers:
* **fs:** Handle file and directory/folder system. (creating a folder, Accessing and open a file, editing a file, copying a file, removing a file or a directory)
* **os:** About computer’s operating system.
* **path:** It includes methods to deal with file paths. Examples: Identify the path of a specific file or folder, identify the path to the root directory and identify the extension of a file
* **http:** It creates an HTTP server in Node.js. Examples: Ability to receive and handle HTTP request and managing connections
* **Events:** used to own and trigger events. Examples: Creating, firing and listening for your own events
* To use, you simply using the require keyword

Example: -

const fs = require('fs') //imports the file system module.

const os = require(‘os’)

const path = require(‘path’)

const http = require(‘http’)

const EventEmitter = require(‘events’)

Building HTTP web server

Web server:

* computer that has an HTTP server software installed on it that helps deliver/handle HTTP requests from client, like your browser, and provided HTTP responses like an HTML page or JSON from a remote server computer.
* Or common subset of an application server, that delivers static web content (Example: HTML pages, files, images, video) in response to HTTP requests from a web browser.

Application server:

* It can deliver web content too, just like any web server. However, its primary job is to enable interaction between browsers and server-side application code to generate and deliver dynamic content.
* But in current practice, most web servers support plug-ins for scripting languages (e.g., PHP) that enable the web server to generate dynamic content and increasing number of application servers incorporate web server capabilities and use HTTP as their primary protocol for interfacing with web servers.

File server:

* The computer that stores the files so that other computers on the same network can access them.

**Building HTTP web server using NodeJS: serving a simple message using HTTP module**

**4.7 Building HTTP web server using Express module: serving our static Apple website with Express**

Express framework:

* It is an open-source server-side framework that written in JavaScript
* Used to build HTTP servers and set up connections.
* Using this connection, data sending and receiving can be done.
* Express is built on top of Node. Js
* built-in HTTP module and is available through the npm
* Express is a framework, not just a module.

**Framework vs library vs package vs module:**

Module:

* set of methods or functions (smallest piece of software) ready to be used somewhere else
* provides a single piece of functionality

Package:

Is a collection of modules of the same functional purpose. Eg. Any package from NPM

Library:

* Is a collection of packages. Eg. Bootstrap

Framework:

* Set of libraries.
* Framework does not just offer functionalities, but it also provides an architecture for the development work. In other words, you do not include a framework, but you integrate your code into it.
* framework forces its coding style on to the developer/user or it is in charge of the flow. It provides some places for you to plug in your code, but it calls the code you plugged in as needed. Eg Express package

**Module 🡪 Package 🡪 Library 🡪 Framework**

Set of methods/fn Set of modules set of packages set of Libraries

**Why Express package is better than the HTTP module to create web servers?**

* HTTP module is an in-build module which is pre-installed along with NodeJS
* Express is installed explicitly using npm command: npm install express.
* Express is a framework as a whole, But HTTP is not a framework as a whole rather it is just a module.
* Express provide express.static function for static page hosting , HTTP does not
* HTTP module provides various tools (functions) to do things for networking like making a server, client, etc. Express along with what HTTP does provide many more functions in order to make development easy.
* Express took the functionality of the HTTP module and added additional functionalities for faster and efficient web development.