Node JS Modules

* A set of functions that you want to include in your applications.
* In Node JS, modules are reusable units of code that encapsulate related functionality and promote modular programming.
* Modules allow you to organize your code into separate files, making it easier to maintain and reuse code across different parts of your applications.

Built-in Modules

* Node.js has a set of built-in modules which you can use without any further installation.

Include module: To include a module, use the require() function with the name of the module:

Syntax: var http = require('module\_name');

Core modules

* Node JS comes with a set of built-in core modules that provides essential functionality. These core modules, such as http, fs, path, and util, can be imported using the require() function without the need for additional installation. They provide a wide range of functionalities for tasks like creating web servers, handling file operations, working with paths, and more.

In Node.js, the core modules are the built-in modules that come with the Node.js runtime environment. These modules provide essential functionality for performing various tasks.

Here is a list of some of the core modules in Node.js:

1. HTTP: Provides functionality to create HTTP servers and make HTTP requests.

2. HTTPS: Similar to the HTTP module, but for creating secure HTTPS servers and making HTTPS requests.

3. FS (File System): Allows interaction with the file system, including reading and writing files, creating directories, etc.

4. Path: Provides utilities for working with file paths, such as resolving paths, extracting file names, etc.

5. OS: Provides operating system-related utility functions, allowing you to retrieve information about the operating system.

6. Events: Enables the implementation of event-driven programming in Node.js, allowing objects to emit and listen to events.

7. Stream: Provides an interface for handling streaming data, allowing you to read from and write to streams efficiently.

8. Crypto: Offers cryptographic functionality, such as creating secure hashes, generating secure random numbers, etc.

9. Process: Provides information and control over the current Node.js process, including accessing command-line arguments, environment variables, etc.

10. Buffer: Allows the manipulation of binary data directly, useful for handling streams, file operations, and network operations.

11. URL: Provides utilities for URL resolution, parsing, and formatting.

12. Query String: Enables parsing and formatting URL query strings.

13. Net: Provides network-related functionality, including creating servers and clients for TCP and UNIX domain sockets.

14. DNS: Offers DNS-related functions, such as DNS lookups and reverse lookups.

15. Cluster: Enables the creation of child processes to take advantage of multi-core systems, improving the performance of Node.js applications.

These are just some of the core modules available in Node.js. There are additional modules as well, and you can also create your own modules or use modules from the npm ecosystem to extend the functionality of your Node.js applications.

Local Modules: Local module are the modules that you create in your own Nodejs project. They are stored in separate file in your project directory and can be acquire using relative path. By splitting your code in multiple modules, you can organize your codebase and improve code reusability.

Third-Party module: Third- Party module are the modules that are created by developers and made available for public use. They are not included within Nodejs , they need to be installed using package manager like(NPM).

These module published on npm registry and can be easily integrated into your project using require function after installation.