1. \*\*http://

The `http` module in Node.js is used to create an HTTP server or make HTTP requests. It allows you to handle incoming HTTP requests and send responses. You can create a server with `http.createServer()` and define request handlers to process incoming requests.

2. \*\*events://

The `events` module provides an EventEmitter class that enables the implementation of the observer pattern. It allows objects to emit and listen for events. You can use it to handle custom events within your application.

3. \*\*filesystem://

The `fs` (filesystem) module allows you to interact with the file system. It provides methods to read, write, and manipulate files and directories. Common operations include reading file contents (`fs.readFile()`), writing to files (`fs.writeFile()`), and managing directories.

4. \*\*console://

The `console` module is used for basic input and output in the terminal. You can use it to print messages, log information, and debug your Node.js applications. Common methods include `console.log()`, `console.error()`, and `console.info()`.

5. \*\*buffer://

The `buffer` module provides a way to handle binary data in Node.js. Buffers are used to represent sequences of binary data, and they are particularly useful when working with binary streams or when dealing with binary protocols.

6. \*\*globals://

The `globals` module doesn't exist in Node.js by default. However, Node.js has a set of global objects and functions that are available in all modules without the need to require them. Examples include `console`, `setTimeout`, and `setInterval`.

7. \*\*stream://

The `stream` module is used for working with streams of data. Streams are an important concept in Node.js, providing an efficient way to handle data flow. You can use readable, writable, and duplex streams to process data in chunks, which is especially useful for large datasets.

8. \*\*url://

The `url` module provides methods for parsing and formatting URLs. You can use it to break down a URL into its components (protocol, hostname, pathname, etc.) or to create a URL from its components.

9. \*\*path://

The `path` module deals with file paths and directory paths. It provides methods to manipulate file paths, such as joining paths, resolving relative paths, and extracting file or directory names.

10. \*\*os://

The `os` module provides information about the operating system. You can use it to get information about the system's architecture, platform, and available memory. It also provides methods to interact with the file system, such as getting the home directory.

11. \*\*process://

The `process` module provides information about the currently running Node.js process. It allows you to interact with the process environment, handle signals, and control the execution of the script. You can access command line arguments, environment variables, and exit the process using methods like `process.exit()`.