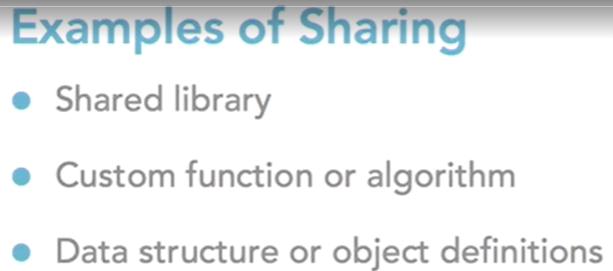
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**LEARNING NODE.JS**

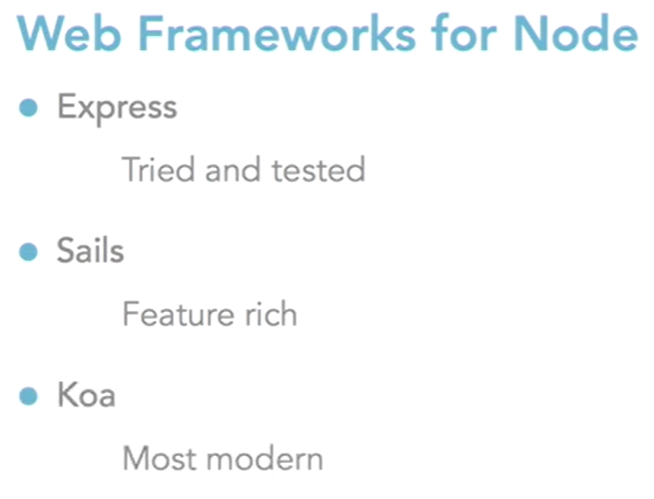
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* **Node** is an open source, cross-platform runtime environment for server-side and networking applications. It’s built on top of Chrome’s Javascript runtime.





* Javascript is a dynamic language, meaning the type is determined by value, not when the variable is declared.
* **Packages** are one or more modules bundled together. One of the most popular is called *Lodash*. To install it, run the command *npm install lodash*.
* There are several different types of **Node Package Manager (NPM)** packages. Some work as command line interfaces (CLIs). *Nodemon* is one of those.
* To run Nodemon it might be necessary to change some settings. Run PowerShell as an Admin and type the command **Get-ExecutionPolicy -List** to see what Execution Policy you have. If the value is “Restricted”, change it with **Set-ExecutionPolicy -ExecutionPolicy RemoteSigned**.
* Using *Nodemon* allows us to automatically execute our .js file anytime there’s a change.
* What if we want to distribute our app or project, or put it into a Git repository? The **package.json** file is the answer. Among other things, it stores a list of the packages you depend on in your project. That way, when using *npm install* it will go through that list and install everything automatically. To create it, type *npm init*.
* *JSON.stringify* is a function that converts data in JSON format.



* **Socket.IO** allows us to notify clients or the app running in the browser when another user has sent a chat message, that way the app will update their message list instantly.
* **Promises** return an object which promises to do some work. This object has separate callbacks for success and for failures.
* In order to work with **Await** we need to declare our function as **Async**.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**NODE.JS ESSENTIAL TRAINING**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* The **Console** object is part of a Global object. This means we can use any of the objects or values that are available to us on the global object within our JavaScript files. The **Global** object contains all of the objects, values, and methods that we can use in a Node.js file without having to import any functionality.
* The **Process** object contains information about the current process as well as tools to allow us to interact with the process. We can get environment information, read environment variables, communicate with the terminal or parent processes through standard input and standard output. The Process object can collect information from the terminal when we load the application.
* When variables are declared as **const** it is because you don’t want the users to be able to change their value. Anytime you want the opposite, declare the variable with **let**.
* The **require** function is what we use to load modules. The modules that you don’t have to install with NPM are called **core modules**.
* **Readline** is a module that we can use to help us build an application that would ask questions of a terminal user. It’s an interface around readable and writable streams that allows us to easily write code that would prompt the user and collect their answers.
* The Require function is part of the **Common JS** module pattern, but it only represents the half that loads the module. The other half is *module.exports*, or the mechanism that we use to export data and functionality from a module.
* The **Event Emitter** implementation of the Pub-Sub design pattern, which gives us a mechanism for emitting custom events and wiring up listeners and handlers for those events.
* The **fs** module can be used to list files in directories, create new files in directories, stream files, watch files, modify file permissions, and so on.
* When reading text files, the encoding needs to be provided in the function.
* Reading files with **streams** causes the application to use less memory because instead of reading everything all at once and loading it into a buffer, you’re reading files bit by bit and chunk by chunk.
* Node comes with a **child** process module which allows you to execute external processes in your environment. Your Node app can communicate with other applications within the environment that is hosted.