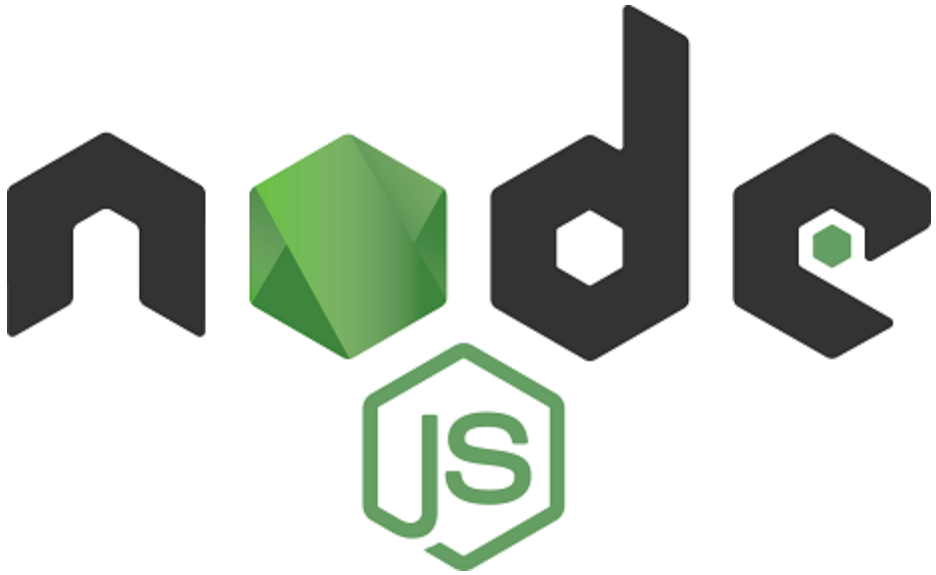
node
JS

Training Agenda

- NodeJS Overview
- Modules System
- File System
- Buffers & Streams
- Event System
- Express
- Data Persistence
- Socket Programming

Prerequisites:

- Working knowledge of HTML, CSS & JavaScript
- Interest to Learn

NodeJS

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

-nodejs.org

NodeJS Process Model

Node.js runs in a single process and the application code runs in a single thread.

All the user requests to web application will be handled by a single thread and all the I/O work or long running job is performed asynchronously for a particular request.

An event loop is constantly watching for the events to be raised for an asynchronous job and executing callback function when the job completes.

Internally, Node.js uses *libev* for the event loop which in turn uses internal C++ thread pool to provide asynchronous I/O.

Module System

Use of imports/exports for importing and exporting the modules in application.

Salient points of the module system:	Each file is its own module.
	Each file has access to the current module definition using the *module* variable.
	The export of the current module is determined by the *module.exports* variable.
	To import a module, use the globally available *require* function.

Require Function

The Node.js *require* function is the main way of importing a module into the current file.

The *require* function blocks further code execution until the module has been loaded.

There is no accidental global scope.

you can choose to call require() based on some condition and therefore load the module only if you need it.

After the first time a require call is made to a particular file, the module.exports is cached.

Module System(Cntd…)

Assume you require('something'). Then the follow is the logic followed by Node.js:

- If something is a core module, return it.
- If something is a relative path (starts with ‘./’ , ‘../’) return that file OR folder.
- If not, look for node_modules/filename or node_modules/foldername each level up until you find a file OR folder that matches something.

When matching a file OR folder:, follow these steps:

- If it matched a file name, return it.
- If it matched a folder name and it has package.json with main, return that file.
- If it matched a folder name and it has an index file, return it.

Few Core Modules

path

os

fs

events

http

utils

Important Globals

Console	the console plays an important part in quickly showing what is happening in your application when you need to debug it.
Timers	setTimeout only executes the callback function once after the specified duration. But setInterval calls the callback repeatedly after every passing of the specified duration.
__filename and __dirname	These variables are available in each file and give you the full path to the file and directory for the current module.
Process	Use the process object to access the command line arguments. Used to put the callback into the next cycle of the Node.js event loop.
Buffer	Native and fast support to handle binary data.
Global	The variable global is our handle to the global namespace in Node

Node Package Manager

- NPM is the eco-system to manage the project dependencies

Few NPM Commands

npm init	Generates package.json file in local project directory
npm install	Install the packages/ dependencies
npm uninstall	Uninstall the packages/dependencies
npm config get/set	Gest /set the npm eco-system
npm update	Update the project dependencies
npm ls	List down the dependencies
npm search	Search the listed package on npm registry
npm outdated	List down the outdated package

Event System

EventEmitter is a class designed to make it easy to emit events and subscribe to raised events.

Subscribing : *built-in support* for multiple subscribers for events.

Unsubscribing : *EventEmitter* has a *removeListener* function that takes an event name followed by a function object to remove from the listening queue.

Memory Leak : A common cause of memory leak is forgetting to unsubscribe for the event.

A number of classes inside core Node.js inherit from EventEmitter.

Buffers & Streams

Streams play an important role in creating performant web applications.

Improvement in user experience and better utilization of server resources is the main motivation behind steams.

All of the stream classes inherit from a base abstract Stream class which in turn inherits from EventEmitter.

All the streams support a pipe operation that can be done using the pipe member function.

Streams (Cntd…)

Readable	A readable stream is one that you can read data from but not write to.
	A good example of this is process. stdin, which can be used to stream data from the standard input.
Writable	A writable stream is one that you can write to but not read from.
	A good example is process.stdout, which can be used to stream data to the standard output.
Duplex	A duplex stream is one that you can both read from and write to.
	A good example of this is the network socket. You can write data to the network socket as well as read data from it.
Transform	A transform stream is a special case of a duplex stream where the output of the stream is in some way computed from the input.
	A good example of these is encryption and compression streams.

Express

Web application framework for Node apps.

To create an express app, make a call require('express')

Express can accept middleware using the 'use' function which can be registered with http.createServer.

Express Request / Response objects are derived from standard NodeJS http Request / Response

Request object can handle URL : Route Parameter & Querystrings

```
npm install express
```

Data Persistence

- Why NoSQL ?
  - Scalability
  - Ease of Development
- NoSQL servers can be placed into four broad categories:
  - Document databases
  - Key-value databases
  - Column-family databases
  - Graph databases

MongoDB

- A MongoDB deployment consists of multiple databases.
- Each database can contain multiple collections.
  - A *collection* is simply a name that you give to a *collection of documents*.
- Each collection can contain multiple documents.
  - A document is effectively a JSON document

```
npm install mongodb
```

Socket Programming

Bi-directional
Full duplex
communication

Continuous
channel of
communication

```
npm install socket.io
```

Scalability

- Ensuring uptime using Process Manager :

```
npm install pm2
```

- Node Clustering utilize all the CPU cores available to us on a multi-core system.
  - New Workers
  - The Ideal Worker
  - Communication with the Master

References

Books :

- Beginning NodeJS by Basarat Ali Syed
- Node.JS Web Development by David Herron

Web :

- https://nodejs.org/en/docs
- https://stackoverflow.com