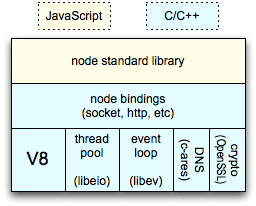
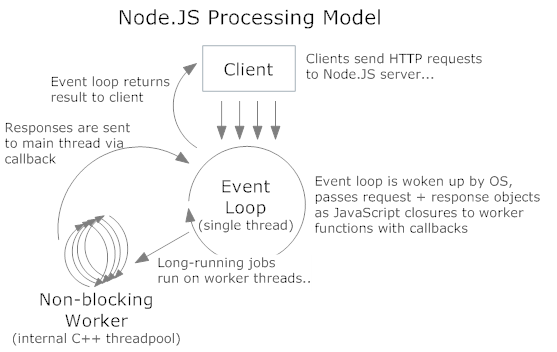
**Introduction to nodejs:**

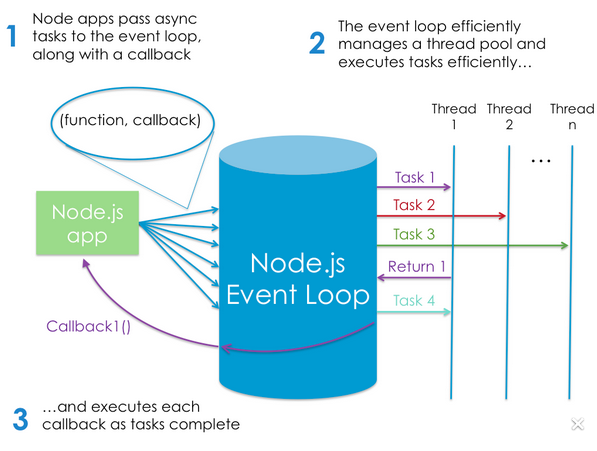
* Node.js is a JavaScript runtime that uses the V8 engine developed by Google for use in Chrome.
* V8 compiles and executes JavaScript at lightning speeds mainly due to the fact that V8 compiles JavaScript into native machine code.
* In addition to lightning fast JavaScript execution, the real magic behind Node.js is the event loop.
* The event loop is a single thread that performs all I/O operations asynchronously.
* Traditionally, I/O operations either run synchronously (blocking) or asynchronously by spawning off parallel threads to perform the work. This old approach consumes a lot of memory and is notoriously difficult to program.
* In contrast, when a Node application needs to perform an I/O operation, it sends an asynchronous task to the event loop, along with a callback function, and then continues to execute the rest of its program. When the async operation completes, the event loop returns to the task to execute its callback.

**NODEJS ARCHITECTURE :**

****

**Event loop:**

****

****

**Closures:**

A closure is an inner function that has access to the outer (enclosing) function’s variables—scope chain.

The closure has three scope chains: it has access to its own scope (variables defined between its curly brackets), it has access to the outer function’s variables, and it has access to the global variables.

The inner function has access not only to the outer function’s variables, but also to the outer function’s parameters.

[**https://developer.mozilla.org/en/docs/Web/JavaScript/Closures**](https://developer.mozilla.org/en/docs/Web/JavaScript/Closures)

**Example:**

function makeFunc() {

var name = "Mozilla";

function displayName() {//this is a closure function

console.log(name);

}

return displayName;

}

var myFunc = makeFunc();

myFunc();

**export keyword:**

it is used to create js file as module and can be used in another js file

<http://www.sitepoint.com/understanding-module-exports-exports-node-js/>

**exportexample.js file**

/\*\*

\* Constructor

\* @param {Object} params

\*/

var exportexample = function(params) {

console.log(JSON.stringify(params)+"checking params");

//console.log(username+"checking username"+username)

if(params.username !=='undefined'){

this.username=params.username;

}

if(params.password !=='undefined'){

this.password=params.password;

}

}

exportexample.prototype.logindetails = function(callback) {

console.log("username from login details:: "+this.username+" :: "+this.password);

callback("details reachec successfully");

};

exports.exportexample = exportexample;

**main.js file**

var Exportexample=require('./exportexample').exportexample;

var myobject={ "username":"krishna",

"password":12345

}

var exportexample =new Exportexample(myobject);

exportexample.logindetails(function(result){

console.log("i am in main.js file "+result)

});

**Callbacks:**

* A callback function is a function that is passed to another function (let’s call this other function “otherFunction”) as a parameter, and the callback function is called (or executed) inside the otherFunction.
* A callback function is essentially a pattern (an established solution to a common problem), and therefore, the use of a callback function is also known as a callback pattern.

**Example:**

var name="krishna";

var place="banglore";

secondfunction(name,place,function(data){

console.log("checking callbackdata::"+JSON.stringify(data))

});

function secondfunction(name,place,mycallback){

var object={};

object.name=name;

object.place=place;

mycallback(object);

}

**Creating pakage.json file:**

Go to root folder of the project and type the below command and enter the required field

**Cmd:** npm init

**Sample file:**

Package.json

{

"name": "sampleproject",

"version": "1.0.0",

"description": "this is sample project",

"main": "samplecallback.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

**Installing or loading the required node modules:**

**Npm :** is a node pakage manager where it download all the required node modules from cdns(module from cloud);

**Syntax:**

npm install <required node module> --save

after the installation we can find “node\_modules” created by nodejs and “fs-extra” module is present in node\_modules.

**Example:**

If we want to do file operation, we have to install “fs” or “fs-extra” node modules

Npm install “fs-extra” –save

Installation of node modules from pakage.json:

**require:** it is used to load node modules AND .js files

syntax to load node modules:

var variable=require(“name of the module”);

EX: var fs=require(“fs-extra”);

**File operations:**

<https://www.npmjs.com/package/node-fs-extra>

**Example:**

var fs = require('fs-extra');

var file = './trash/content.txt';

var data="i am in banglore";

fs.outputFile(file,data, function(err) {

console.log(err); //null

console.log("i am in output file");

})

**Async:**

**async module** is used to control the execution of the code

**asyn.series :** is used to execute the code in synchronous way. i..e step by step.but we can not carry forward the data to next functions

<https://www.npmjs.com/package/async>

**syntax:**

async.series([

function(callback1){

//do stuff

Callback1

},

function(callback2){

//do stuff

Callback2

}

],function(){//this is optional call back

//end of series

})

**Async.waterfall:**

Functionality is same as async.serice but in async.waterfall we can carry forward the data to next consequent function

<https://www.npmjs.com/package/async-waterfall>

**syntax:**

async.waterfall([

function(callback){

callback(null, 'one', 'two');

},

function(arg1, arg2, callback){

console.log("second function "+arg1+"::"+arg2)

callback(null, 'three');

},

function(arg1, callback){

// arg1 now equals 'three'

console.log("third function "+arg1)

callback(null, 'done');

}

], function (err, result) {

// result now equals 'done'

console.log("end of water fall");

});

Task1:

create a javascript class in a file and export it .

so that it can be used in another js files by createing object of exported file and asscees the methods

Task2::

create a callback function and pass the parameter.

if parameter reached to callback,send back message "parames received "

Task3:

create a callback function and pass the paramter to callback function.

prepare json object in callback function with the received paramters and send back json object back.

print the json object.

Task4

create a text file using "fs-extra" node module and write the following json content.

var jsoncontent={

"name" : "krishna",

"place":"banglore",

"qualification" : "b.tech"

}

Task5:

create a text file in a specific folder .

read the file from the folder.

empty the folder after reading the file.

Task6:

create a file and write the content in the file.

read the file and display the content.

read and write operation should be done in synchronous way

Task 7:

read the data from the file and send the data to another function and display

Hint:use async.waterfall

async.waterfall([

function(callback){

//read the data from file here

callback(null, filedata);

},

function(filedata, callback){

//print the file data here

callback(null, "finished");

}

], function (err, result) {

// result now equals 'finish'

console.log("end of water fall");

});

**Task 8:**

Create an array and iterate array using async.each method and print the values