Course Title: Node.js

Faculty: Ajay Reddy

L&T Technology Services

LTTS GLOBAL ENGINEERING ACADEMY

Agenda

Agenda

8

9

10

11

12

13

14

Version

Version	Reviewed by	Approved by	Remarks
1.0	Ajay Reddy		

Node.js Syllabus of the Couse on this slide

Learning Outcome of the course

- Understand outcome of the how node.js act as server.
- Understand how node.js giving response to the client .
- Able to create and read the files as well as and upload the file using express framework.
- Understand how we are connect with Mysql OR MongoDB database
- Understand the concepts how node.js executes basic operations in command prompt.
- Knows about how Javascript work in server end.

Pre-Requisites of the course

- Basic Knowledge on Javascript Concepts
- Basic fundamentals on HTML tags.
- Understand the basic IDE environments like Visual Studio IDE with few commands
- Node.js installation is required(Using VS_Code)
- Understand basic knowledge on browser applications
- Understand basic knowledge on http protocol

Introduction about Node.js

Introduction about Node.Js

- Node.js is an open source server environment
- Developed by Ryan Dahl(Open Js Foundation) in year of 2009.
- Written in c, C++, Javascript code.
- Node.js is free to download in Window Operating system or Linux or Unix.
- Node.js runs on various platforms (Windows, Linux, Unix, Mac OS X, etc.)
- Node.js uses JavaScript on the server

Why Node.js?

Why Node.js

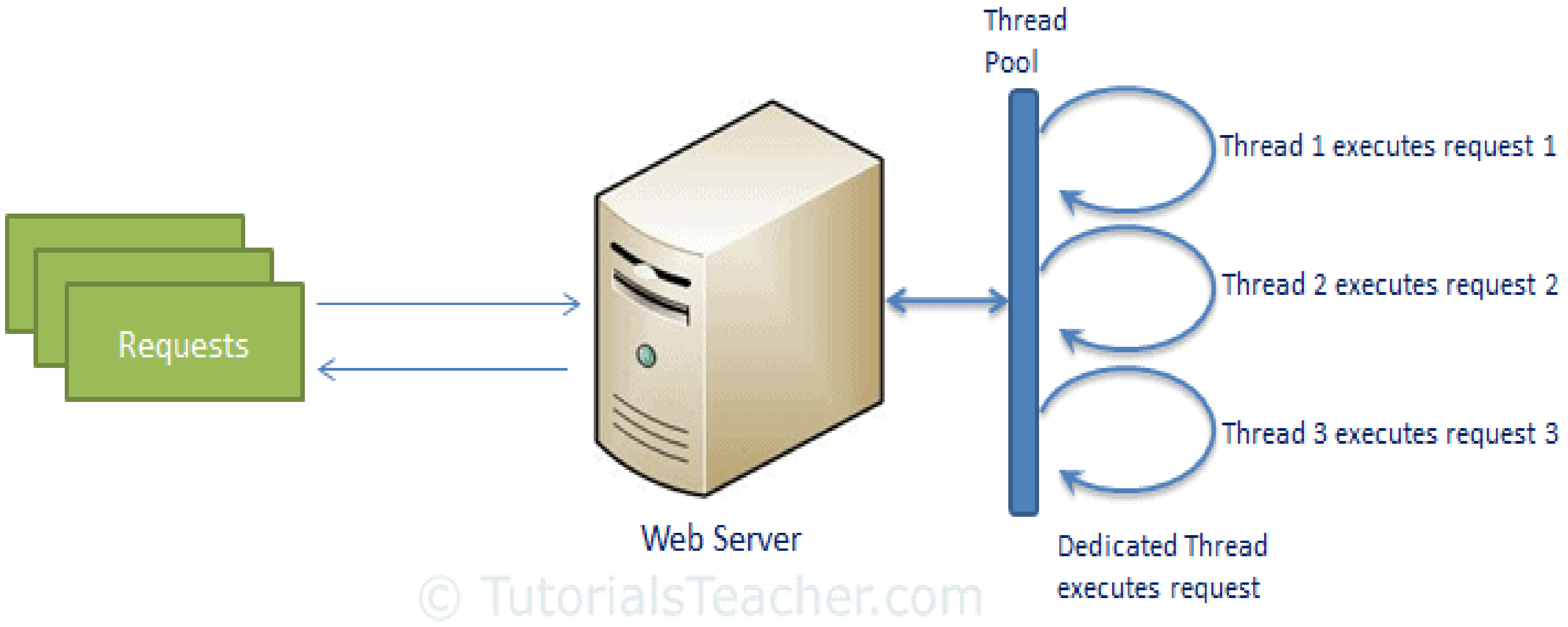
- Node.js can generate dynamic page content
- Node.js can create, open, read, write, delete, and close files on the server
- Node.js can collect form data
- Node.js can add, delete, modify data in your database
- Node.js can upload file in the system
- Install other framework technologies (express)

MEAN(MONGODB, EXPRESS.JS,ANGULAR,NODE.JS)

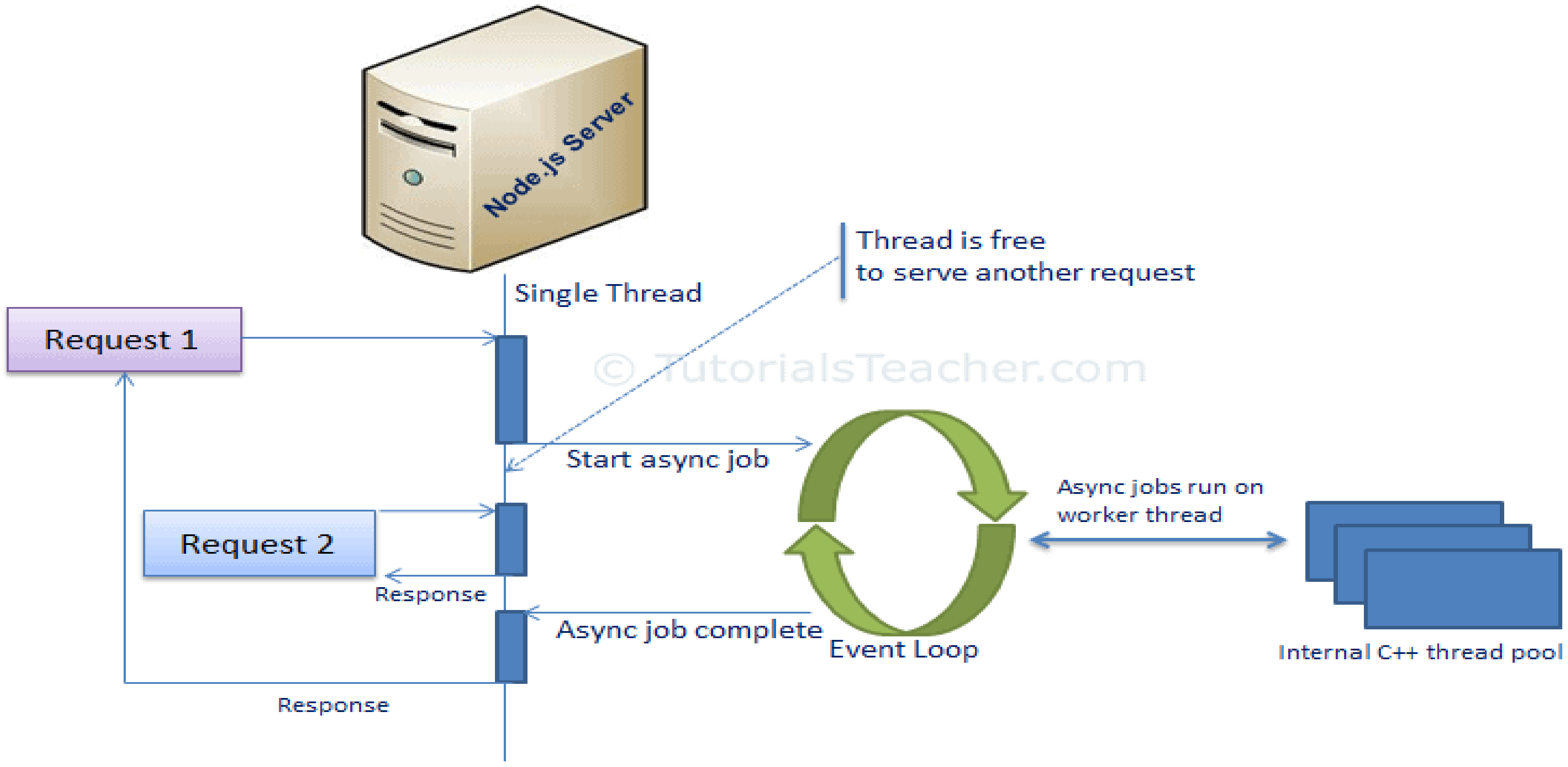
MERN(MONGODB,EXPRESS.JS,REACTJS,NODE.JS)

How Node.js work

PHP or Asp Working Mechanism


Thread Pool
Thread 1 executes request 1
Thread 2 executes request 2
Thread 3 executes request 3
Requests
Web Server
Dedicated Thread executes request
© TutorialsTeacher.com

Node.js Working Mechanism



How Node.js works:

- Send the task to the computer's file system.
- Ready to handle the next request.
- When the file system has opened and read the file, the server returns the content to the client.

*****Node.js eliminates the waiting, and simply continues with the next request.

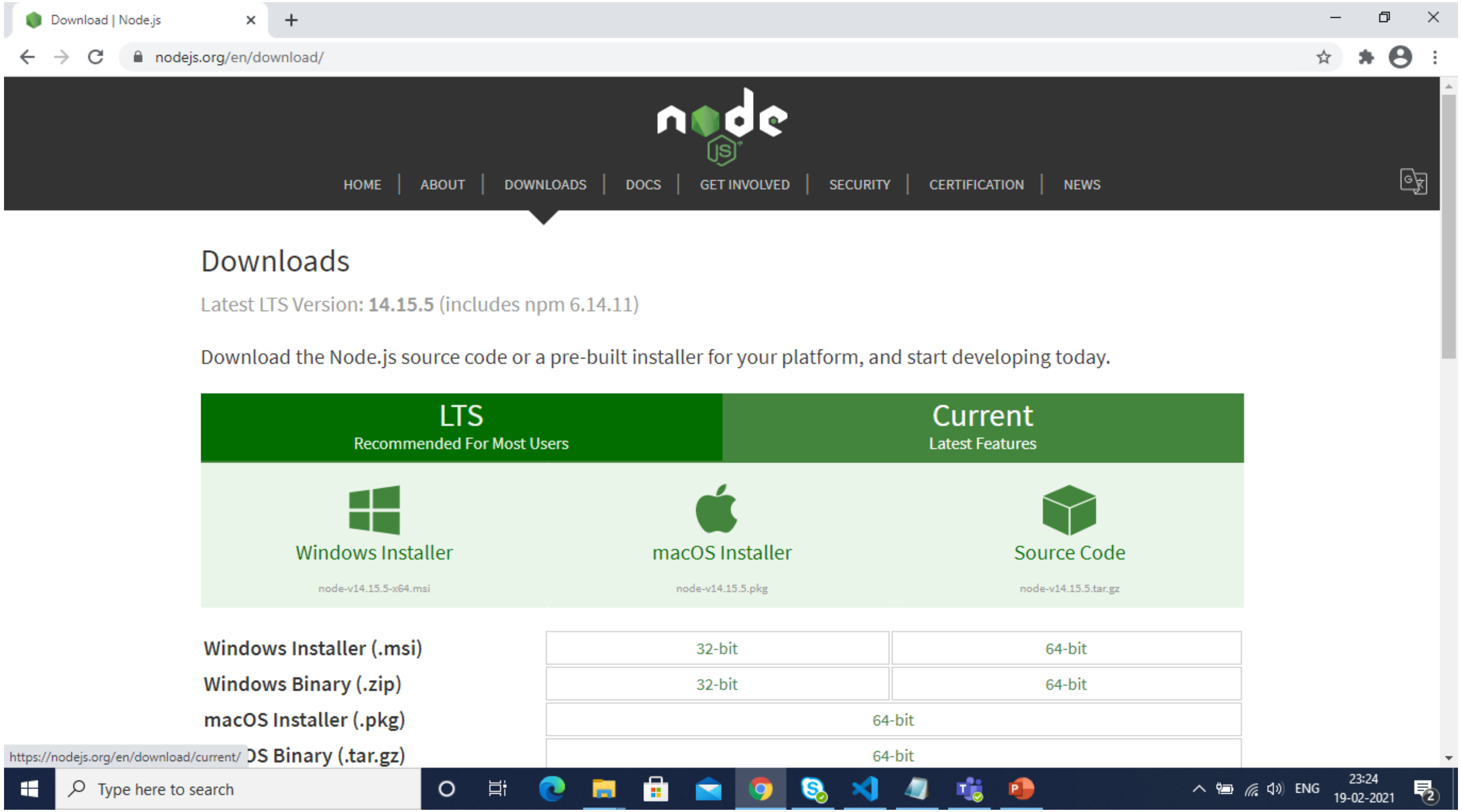
Node.js runs single-threaded, non-blocking, asynchronously programming, which is very memory efficient

Advantages Over PHP Or ASp

Advantages Over PHP or Asp

- Sends the task to the computer's file system.
- Waits while the file system opens and reads the file.
- Returns the content to the client.
- Ready to handle the next request.
- PHP or Asp works runs Multi-threaded, blocking, synchronous programming, which is very inefficient memory .

Down and Install Node.js



Downloads

Latest LTS Version: **14.15.5** (includes npm 6.14.11)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features

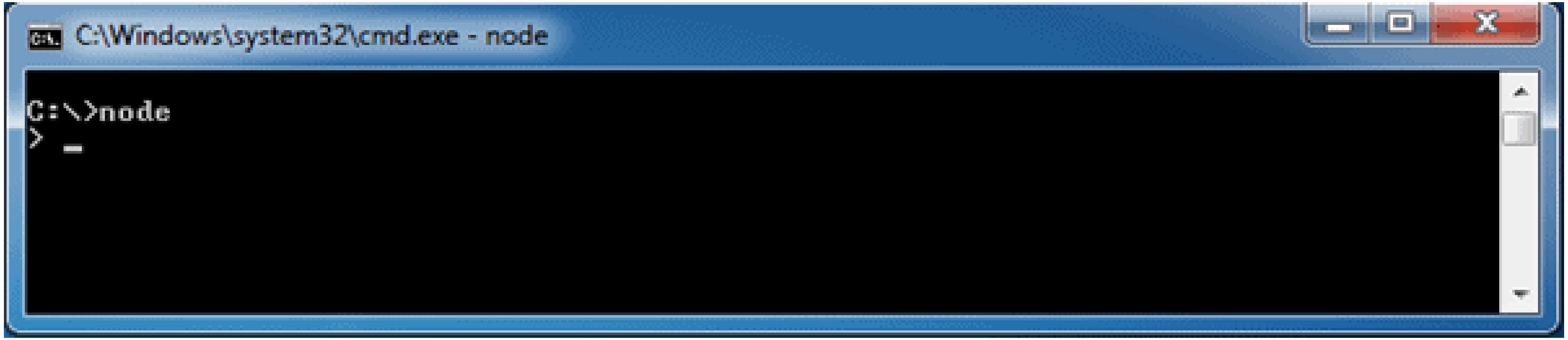
Windows Installer	macOS Installer	Source Code
node-v14.15.5-x64.msi	node-v14.15.5.pkg	node-v14.15.5.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
DS Binary (.tar.gz)	64-bit	

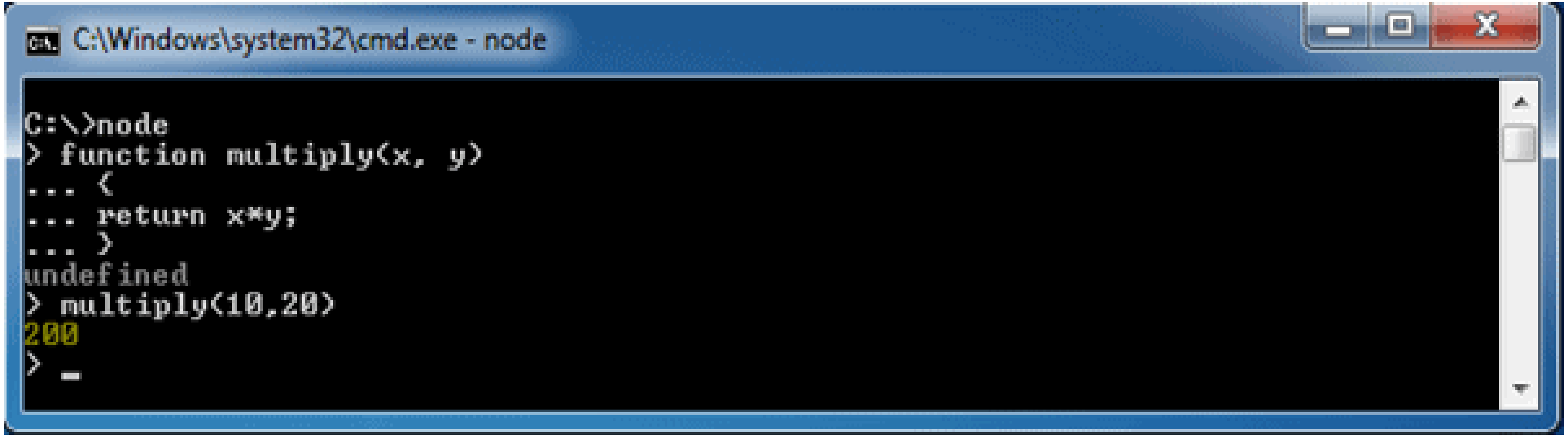
Check Version of Node & NPM

- `Npm install npm -g`
- `Npm -v`
- `Node -v`
- `Npm install express --save`
- `Npm install -g express`

Node.js on Console



```
C:\>node
> _
```



```
C:\>node
> function multiply(x, y)
... {
... return x*y;
... }
undefined
> multiply(10,20)
200
> _
```

First Program on Node.js

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Hello World !');
  res.write("Giving data from the server");
  res.end();
}).listen(5100);
```

Then after goto browser window and type

```
   localhost:5100
```

Module in Node.js

Module in Node.js

***What is a Module in Node.js?

Consider modules to be the same as JavaScript libraries.

A set of functions you want to include in your application.

****Include Modules

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

```
var http = require('http');
```

Modules in Node.js

Node.js is a light weight framework. The core modules include bare minimum functionalities of Node.js. These core modules are compiled into its binary distribution and load automatically when Node.js process starts. However, you need to import the core module first in order to use it in your application.

Core Module	Description
http	http module includes classes, methods and events to create Node.js http server.
url	url module includes methods for URL resolution and parsing.
querystring	querystring module includes methods to deal with query string.
path	path module includes methods to deal with file paths.
fs	fs module includes classes, methods, and events to work with file I/O.
util	util module includes utility functions useful for programmers.

Own Module in Node.js

test.js

```
exports.myDT = function () {
    return Date();
  };
```

Sample.js

```
var http = require('http');
var dt = require('./test');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write("The date and time is: " + dt.myDT());
  res.end();
}).listen(5100);
```

Built-in Module in Node.js

The Built-in HTTP Module

Node.js has a built-in module called HTTP, which allows Node.js

to transfer data over the Hyper Text Transfer Protocol (HTTP).

To include the HTTP module, use the require() method:

Ex:
```
var http = require('http');
```

Node.js WebServer

Node.js as a Web Server

The HTTP module can create an HTTP server that listens to server ports and gives a response back to the client.

Use the createServer() method to create an HTTP server:

```
var http = require('http');


//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```

HTTP Header

Http Header

If the response from the HTTP server is supposed to be displayed as HTML, you should include an HTTP header with the correct content type:

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write('Hello World!');
  res.end();
}).listen(8080);
```

The first argument of the res.writeHead() method is the status code,

200 means that all is OK, the second argument is an object containing the response headers.

Node.js Query String

Node.js Query String

The function passed into the http.createServer() has a req argument

that represents the request from the client, as an object

(http.IncomingMessage object).

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.write(req.url);
  res.end();
}).listen(5100);
```

Note: localhost:5100/ ajay →/ajay

Node.js File System

Node.js File System

Node.js as a File Server

The Node.js file system module allows you to work with the file system on your computer.

To include the File System module, use the require() method:

Common use for the File System module:

Read files

Create files

Update files

Delete files

Rename files

Read File

The fs.readFile() method is used to read files on your computer.

Assume we have the following HTML file (located in the same folder as Node.js):

demo.html

```
<html>
<body>
<h1>File Read </h1>
<p>My paragraph.</p>
</body>
</html>
```

Read File

Sample.js

```
var http = require('http');
var fs = require('fs');
http.createServer(function (req, res) {
  fs.readFile('demo.html', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(5100);
```

Create File: fs.appendFile()

- The File System module has methods for creating new files:

```
fs.appendFile()
fs.open()
fs.writeFile()
```

The fs.appendFile() method appends specified content to a file. If the file does not exist, the file will be created:

Create a new file using the appendFile() method:

```
var fs = require('fs');
fs.appendFile(‘test1.txt', 'Hello append!', function (err) {
 if (err) throw err;
 console.log('Saved!');
});
```

Create File: fs.Open()

- The fs.open() method takes a "flag" as the second argument,
- if the flag is "w" for "writing", the specified file is opened for writing.
- If the file does not exist, an empty file is created:
- Create a new, empty file using the open() method:

```
var fs = require('fs');
fs.open(‘test1.txt', 'w', function (err, file) {
 if (err) throw err;
 console.log('Saved!');
});
```

Create File: fs.writeFile()

The fs.writeFile() method replaces the specified file and content if it exists.

If the file does not exist, a new file, containing the specified content,

will be created:

Create a new file using the writeFile() method:

```
var fs = require('fs');
fs.writeFile('test1.txt', 'I'm writing!', function (err) {
  if (err) throw err;
  console.log('Saved!');
});
```

File Uploads

File Uploads:

- To upload File into the Node.js server we need to follows the given steps.

1. Install npm package

```
D:\Nodetest> npm install -g express
D:\Nodetest> npm install express –save
```

2 Install express-fileupload

```
D:\Nodetest> npm install express-fileupload
```

1. Create folder (“D:\Nodetest\upload”)

File Uploads: Sample.js

```
var express=require("express"),
app=express(),
http=require("http").Server(app).listen(5100),
upload=require("express-fileupload");
app.use(upload())
console.log("Server started")
app.get("/",function(req,res)
{     res.sendFile(__dirname+"/index.html")
})
app.post("/", function(req,res)
{     if(req.files)
      {           var file=req.files.filename,
            filename=file.name;
            file.mv("./upload/"+filename,function(err)
            {           if(err)  {                   console.log(err)
                        res.send("error")
                  }
                  else{
                        res.send("Done!!!")
                  }
            })
      }
})
```

File Uploads: index.html

Index.html

```
<h1>
 Hello
</h1>
<form method="post" enctype="multipart/form-data" action="/">
<input type="file" name="filename">
<input type="submit" value="upload">
</form>
```

NPM(Node Package Manager)

NPM

NPM is a package manager for Node.js packages, or modules if you like.

www.npmjs.com hosts thousands of free packages to download and use.

The NPM program is installed on your computer when you install Node.js

What is a Package?

A package in Node.js contains all the files you need for a module.

Modules are JavaScript libraries you can include in your project.

Ex: npm install express

Node.Js with MySql

Mysql :

- To access a MySQL database with Node.js, you need a MySQL driver. This tutorial will use the "mysql" module, downloaded from NPM.
- To download and install the "mysql" module, open the Command Terminal and execute the following:

```
C:\Users\Your Name>npm install mysql
```

- Now you have downloaded and installed a mysql database driver.
- Node.js can use this module to manipulate the MySQL database:

```
var mysql = require('mysql');
```

MySql: Create Database

-
  ```
  var mysql = require('mysql');

  var con = mysql.createConnection({
    host: "localhost",
    user: "yourusername",
    password: "yourpassword"
  });

  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    con.query("CREATE DATABASE mydb", function (err, result) {
      if (err) throw err;
      console.log("Database created");
    });
  });
  ```

MySql: Create Table

-
  ```
  var mysql = require('mysql');

  var con = mysql.createConnection({
    host: "localhost",
    user: "yourusername",
    password: "yourpassword",
    database: "mydb"
  });

  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = "CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("Table created");
    });
  });
  ```

MySql: Create Table

-
  ```
  var mysql = require('mysql');

  var con = mysql.createConnection({
    host: "localhost",
    user: "yourusername",
    password: "yourpassword",
    database: "mydb"
  });

  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = "CREATE TABLE customers (name VARCHAR(255), address
  VARCHAR(255))";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("Table created");
    });
  });
  ```

MySql: Insert Table

-
  ```
  var mysql = require('mysql');

  var con = mysql.createConnection({
    host: "localhost",
    user: "yourusername",
    password: "yourpassword",
    database: "mydb"
  });

  con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = "INSERT INTO customers (name, address) VALUES
  ('Vijay', 'banglore')";
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("1 record inserted");
    });
  });
  ```

Mysql : Select

```
var mysql = require('mysql');
var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "root",
  database: "ajay"
});
con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM employee", function (err, result, 
fields) {
    if (err) throw err;
    console.log(result);  });   });
```

Node.Js with MongoDB

MongoDB :

- To download and install the official MongoDB driver, open the Command Terminal and execute the following:

Download and install mongodb package:

- C:\Users\Your Name>npm install mongodb

- Now you have downloaded and installed a mongodb database driver.
- Node.js can use this module to manipulate MongoDB databases:

```
var mongo = require('mongodb');
```

MongoDB: Creating Database

- To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct ip address and the name of the database you want to create.
- MongoDB will create the database if it does not exist,and make a connection to it.

Create a database called "mydb":

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

MongoDB: Create Collection (Table)

-
  ```
  var MongoClient = require('mongodb').MongoClient;
  var url = "mongodb://localhost:27017/";

  MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("mydb");
    dbo.createCollection("customers", function(err, res) {
      if (err) throw err;
      console.log("Collection created!");
      db.close();
    });
  });
  ```

Thank You !

L&T Technology Services