Below is a detailed explanation and code for each lab. I'll cover the major points in the question to ensure everything is addressed. Let's go through each of them step by step.

**Lab 01: Create a Node.JS Environment**

1. **Installation of Node.js:**
   * First, install Node.js from the official website: [Node.js Download](https://nodejs.org/).
   * Once installed, you can verify the installation using the following commands in the terminal:

bash

Copy

node -v # check the node version

npm -v # check the npm version

1. **Test through the Node REPL Shell:** You can test Node.js by running the Node REPL shell. Open the terminal and type node, then press Enter. This opens an interactive JavaScript shell where you can run JavaScript code.

Example:

bash

Copy

node

> console.log("Hello from Node.js!");

Hello from Node.js!

1. **Install prompt-sync Module:** The prompt-sync module can be installed via npm. This module allows you to take input from the user synchronously.

Command:

bash

Copy

npm install prompt-sync

1. **Test prompt-sync with Console Module:** Here's an example of using prompt-sync with the console to ask for user input:

javascript

Copy

const prompt = require('prompt-sync')();

const name = prompt("Enter your name: ");

console.log(`Hello, ${name}!`);

**Lab 02: Create a Custom Date Module**

1. **Create Node Package Module myDate (Without package.json):**

Create a file named myDate.js:

javascript

Copy

// myDate.js

module.exports = function() {

return new Date().toString();

};

1. **Create Node Package Module myDate (With package.json):** First, initialize a new Node.js project (if you haven't already):

bash

Copy

npm init -y

Then create the module inside myDate.js:

javascript

Copy

// myDate.js

module.exports = function() {

return new Date().toString();

};

Your package.json file will look like this:

json

Copy

{

"name": "myDate",

"version": "1.0.0",

"description": "A custom date module",

"main": "myDate.js",

"scripts": {

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

},

"keywords": ["date", "module"],

"author": "",

"license": "ISC"

}

1. **Install and Use the Custom Module:** You can install your module with npm locally and use it in your main file:

bash

Copy

npm install

Usage in another file:

javascript

Copy

const myDate = require('./myDate');

console.log("Current date and time: ", myDate());

**Lab 03: Create Node.js Application with Folder Structure**

1. **With Visual Studio Code (VSCode) Framework:** Open VSCode and create a new folder for the project. Inside the folder, create index.js:

javascript

Copy

// index.js

console.log("Welcome to the Node.js Application!");

Run the app:

bash

Copy

node index.js

1. **Without VSCode Framework:** You can still create the same structure and test using the command line. Create the same index.js file and run node index.js.

**Lab 04: Create Angular CLI Application**

1. **Install Angular CLI:**

bash

Copy

npm install -g @angular/cli

1. **Create a New Angular App:**

bash

Copy

ng new hello-world

cd hello-world

ng serve

1. **Create a Class Component:** Inside the src/app folder, create a component using Angular CLI:

bash

Copy

ng generate component hello-world

1. **Define Inline Templates:** Edit the hello-world.component.ts file:

typescript

Copy

@Component({

selector: 'app-hello-world',

template: '<h1>Hello, World!</h1>',

styleUrls: ['./hello-world.component.css']

})

**Lab 05: Create Angular CLI Application with Class Constructors**

1. **Create a Class Constructor with Current Date:** Create a class in your Angular component:

typescript

Copy

export class HelloWorldComponent {

currentDate: Date;

constructor() {

this.currentDate = new Date();

}

}

**Lab 06: Create Angular CLI Application with Expressions and Filters**

1. **Create Angular Expressions and Filters:** Inside the component, you can use expressions like this:

typescript

Copy

<h1>{{ currentDate | date:'short' }}</h1>

This will format the currentDate using Angular's built-in date pipe.

**Lab 07: Create Angular Application with Data Binding**

1. **Create Menu with Directives:** Use ngModel for two-way binding:

html

Copy

<input [(ngModel)]="num1" placeholder="Enter first number">

<input [(ngModel)]="num2" placeholder="Enter second number">

<button (click)="add()">Add</button>

1. **Create Student Form with ngForm and ngSubmit:**

html

Copy

<form (ngSubmit)="onSubmit()" #studentForm="ngForm">

<input type="text" name="name" ngModel required placeholder="Name">

<button type="submit">Submit</button>

</form>

**Lab 08: Create Node.js Application with URL Module**

1. **Decompose URL Components:** Use the url module to parse a URL:

javascript

Copy

const url = require('url');

const urlString = 'http://user:pass@host.com:80/resource/path?query=string#ha';

const parsedUrl = url.parse(urlString);

console.log(parsedUrl);

1. **Resolve the URL:**

javascript

Copy

const resolvedUrl = url.resolve('http://example.com/path', '/newpath');

console.log(resolvedUrl);

**Lab 09: HTTP Server and Client in Node.js**

1. **Create an HTTP Server:**

javascript

Copy

const http = require('http');

http.createServer((req, res) => {

res.writeHead(200, {'Content-Type': 'text/html'});

res.write('<h1>Hello, World!</h1>');

res.end();

}).listen(8080);

console.log('Server running at http://localhost:8080/');

1. **HTTP Client Example:**

javascript

Copy

const http = require('http');

http.get('http://localhost:8080', (res) => {

let data = '';

res.on('data', (chunk) => {

data += chunk;

});

res.on('end', () => {

console.log(data);

});

});

**Lab 10: Simple Arithmetic Operations Form with Express**

1. **Create Arithmetic Operations with Express:** Install Express:

bash

Copy

npm install express

Create a server that handles GET and POST requests for addition:

javascript

Copy

const express = require('express');

const app = express();

app.use(express.urlencoded({ extended: true }));

app.get('/add', (req, res) => {

res.send(`

<form method="POST">

<input type="number" name="num1" />

<input type="number" name="num2" />

<button type="submit">Add</button>

</form>

`);

});

app.post('/add', (req, res) => {

const result = parseInt(req.body.num1) + parseInt(req.body.num2);

res.send(`Result: ${result}`);

});

app.listen(3000, () => {

console.log('Server running on port 3000');

});

**Lab 11: Simple Login Form Page Application with Express**

1. **Create a Login Form:** Use Express to handle the login form:

javascript

Copy

app.get('/login', (req, res) => {

res.send(`

<form method="POST">

<input type="text" name="username" placeholder="Username" />

<input type="password" name="password" placeholder="Password" />

<button type="submit">Login</button>

</form>

`);

});

app.post('/login', (req, res) => {

const { username, password } = req.body;

if (username === 'admin' && password === 'password') {

res.send('Login Successful');

} else {

res.send('Invalid Login Credentials');

}

});

**Lab 12: MongoDB Server with Node.js**

1. **MongoDB Configuration:** Install MongoDB and the MongoDB Node.js driver:

bash

Copy

npm install mongodb

1. **Connecting MongoDB to Node.js:** Example code to insert and retrieve a document:

javascript

Copy

const { MongoClient } = require('mongodb');

async function run() {

const client = new MongoClient('mongodb://localhost:27017');

await client.connect();

const db = client.db('school');

const collection = db.collection('students');

// Insert a student record

await collection.insertOne({ name: 'John Doe', age: 21 });

// Fetch the student record

const student = await collection.findOne({ name: 'John Doe' });

console.log(student);

client.close();

}

run();

**Conclusion:**

Each lab introduces different aspects of Node.js and Angular. The code samples given in this guide allow you to set up basic applications and test fundamental concepts, such as handling HTTP requests, creating custom modules, working with databases, and building Angular components.