Experiment: 1

Aim: Download and Install Node.js and NPM.

Procedure:

1. Downloading the Node.js installer from node.js.org/en/download
2. choosing 64-bit version.zip file.
3. Extract files into location of choice and start the installer.
4. Node installer can be used to download both Node.js and npm on the system.

Downloading npm

5. Download npm in Command line

```
> npm install -g npm
```

After installing Node.js and npm check the installed version using

```
> node -v
  v 18.14.2
> npm -v
  9.5.0
```

6. Installing prompt-sync for performing input output operations dynamic programming in javascript.

```
> npm install prompt sync:
  0 Vulnerabilities.
```

7. Adding Node js to path

Go to environment variables → System Variables → path → edit → new

Add: c:\Program Files\nodejs

80. To start your first program enter in command line:
> npm init.

Experiment: 2

Aim: To write a program to implement the HelloWorld server with HTTP Node.js module.

Procedure:

1. Write your code in a .js file and store in folder with file_name.js
2. Go to command line, go to the file location and type 'node file name.js'
3. A port number is displayed.
4. Open a search Engine, type local host: 3000
5. Output on server

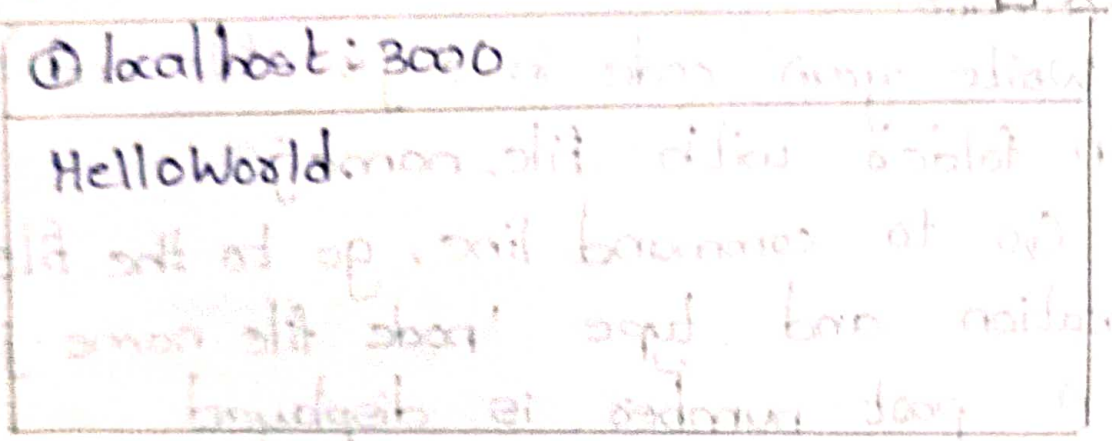
Source code:

In Command line: npm install express

```
// In notepad: importing express module
const express = require ("express");
const app = express();
// Creating express object
  app.get ("/", (req, res) => {
// Listening to get request
  res.send ("<h1> Hello world </h1>");
  });
// sending response to client (html hen)
  app.listen (3000, () => {
  console.log ("listening to port 3000");
  }; // printing
```

// Configuring system to listen to port 3000
// no other server must be running at
that instant.

output:



Experiment: 3

Aim: To write a program to create Calculator Node.js Module with functions add, subtract, multiply and use the module in another Node.js file.

Source code:

```
// drivecal.js
function add(a,b){
  console.log('addition:'+(a+b))
}
function subtract(a,b){
  console.log('subtraction:'+(a-b))
}
function multiply(a,b){
  console.log('multiplication:'+(a*b))
}
function divide(a,b){
  console.log('division:'+(a/b))
}
module.exports{add, subtract, multiply, divide}

// call.js
var drivecal = require('./drivecal')
// importing driver file with functions (require)
const prompt = require('prompt-sync')({sigint = true})
var a = Number(prompt('enter 1st no:'))
var b = Number(prompt('enter 2nd no:'))
console.log("1.add \n 2.subtract \n 3.multiply \n 4.divide \n")
```

```
Var(choice)
{ case 1:
    drive cal.add(a,b);
    break;
  case 2:
    drivecal.subtract(a,b);
    break;
  Case 3:
    drivecal.multiply(a,b);
    break;
  case 4:
    drivecal.divide(a,b);
    break;
}
```

Output:

enter 1st no: 4
enter 2nd no: 5

1. Add
2. Subtract
3. Multiply
4. divide

choice: 3

20

Experiment: 4

Aim: To write a Node.js for the system to perform the following

(i) Create a file

procedure

There are four different files executed with each task.

Source code:

```
// include node fs module
var fs = require("fs");
// (file name, content, call back functions)
fs.writefile('newfile.text', 'my name is Dharani' function (err) {
  if (err) throw err;
console.log("File Created Successfully");
});
```

(ii) read a file

procedure:

Using readfile() function to see location and that type.

Source code:

```
Const.fs = require('fs')
fs.readFile('\users\npl\Desktop mst s22/
    newfile.txt', 'utf 8' (err, data) => {
    if (err){
        console.error(err);
        return;
    }
```

```
        console.log(data);
});
```

(iii) Write a file

Procedure:

writefile() function used in the same fashion as creating a file.

Source code:

```
var fs= require('fs');
const content: 'writing in a file my name is Dharani'
fs.writefile('newFile.txt', content, function.(err){
    if(err) throw err;
    console.log('file written successfully');
});
```

(iv) Deleting a file.

Procedure:

unlink() function is used to delete a file in javascript. It has following two parameters (function_name, error.func())

Procedure:

```
var fs = require('fs');
fs.unlike('new file.txt', function(err){
if(err) throw err;
 console.log('file deleted \');
```

Output:

```
> node create.js
File created successfully
> node write.js
File is written successfully
> node read.js
My name is Dharani
> node delete.js
File deleted.
```

Exercise: 5

Aim: To create and manage an employee database using node.js.

Description: In this program we will create a employee database and we insert data into the database. We create functions like create_db, insert_db, delete_db, drop_db to create, insert, delete and drop the data from the database.

Source code:

```
Var mysql = require ("mysql");
Var con = mysql.CreateConnection ({
            host: "localhost",
            user: "root",
        password: "",
        database: "mydb"});
function create_tb() {
    con.connect (function (err){
        if(err) throw err;
        console.log ("connected!");
    Var sql = "CREATE TABLE student(
ROLL_NO VARCHAR(10), Marks int);";
    con.query (sql, function (err, result){
            if (err) throw err;
            console.log (" Table created");
    });
});
}
function insert_row (roll, marks) {
    con.connect (function (err){
        if(err) throw err;
```

```
        console.log("connected!");
var sql = "INSERT INTO students
                VALUES('${roll}', ${marks})";
    con.query(sql, function(err, result){
        if(err) throw err;
        console.log("1 record inseated");
    });
});
function delete_rec(roll){
    con.connect(function(err){
        if(err) throw err;
        var sql = "DELETE FROM students
        WHERE ROLL NO = '${roll}'";
    con.query(sql, function(err, result){
        if(err) throw err;
        console.log("No of records deleted:"+
    });                    result.affectedRows);
});
function drop_tb(){
    con.connect(function(err){
        if(err) throw err;
        var sql = "DROP TABLE student";
        con.query(sql, function(err, result){
            if(err) throw err;
            console.log("Table deleted");
```

});

});

create_tb();

Exercise: 6

Aim: Implement the following in Angular JS.

a) Angular JS data binding
b) Angular JS directives and Events.
c) Using Angular JS fetching data from MY SQL

a) Angular JS data binding:

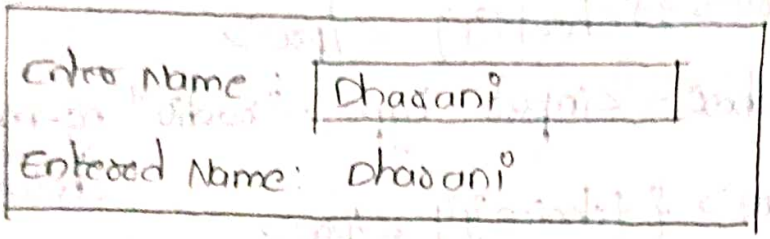
Description: Data binding in Angular JS is the synchronization between the model and the view. Angular JS application usually have a data model. The data model is a collection of data available for the application.

Source code:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>
Angular JS two Binding Example.
</title>
<Script src="http://ajax.googleapis.com/ajax/libs
/angularjs/1.4.8/angular.min.js"></script>
<script type="text/javascript">
var app= angular.module("angulartwobindapp",[]);
app.controller('angulartwobindingctrl', function($scope)
{
$scope.name = 'welcome to Tutlance.com';
});
</script>
</head>
<body ng-app="angulartwobindapp">
<div ng-controller="angulartwobindctrl">
<Enter name:<input type="text" ng-model=
```

```
"name" style="width:250px"/>
<p> Entered Name = {{name}} </p>
</div>
</body>
</html>
```

Output:




b) Angular JS directives and Events.

Source code:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/
        libs/angularjs/1.6.9/angular.min.js">
</script>
<style>
.column {
    float: left;
    text-align: left;
    width: 40%;
}
.row {
    content: "";
    display: table;
}
</style>
<body ng-app="myApp" ng-controller="myContro
                                    -ller">
<h4> Input Box </h4>
<div class = "row">
    <div class = "column">
        Name - <input type="text" ng-model:
```

```
"name">
(<pre> {{name}} </pre>)»
    <pre>{{name}} </pre>
    checkbox - <input type = "checkbox"
                ng-model = "check">
    <pre> {{check}} </pre>
    Radiobox - <input type = "radio" ng-model =
                                    "choice">
    <pre> {{choice}} </pre>
    Number - <input type = "number" ng-model = "num">
    <pre> {{num}} </pre>
    Email - <input type = "email" ng-model = "mail">
    <pre> {{mail}} </pre>
    url - <input type = "url" ng-model = "url">
    <pre> {{.url}} </pre>
    </div>
    <div class = "column">
    Date:
    <input type = "data" ng-model = "date1"
                (change) = "log(date1)">
    <pre> Todays date: {{date+1}} </pre>
    Datetime-local - <input type = "datetime-
                local" ng-model = "date2">
    <pre> {{date2+1}} </pre>
    Time - <input type = "time" ng-model = "mon">
    <pre> {{ time1+1}} </pre>
    Month - <input type = "month" ng-model = "mon">
    <pre> {{ mon+1}} </pre>
    week - <input type = "week" ng-model = "we">
    <pre> {{we+1}} </pre>
```

```
    </div>
  </div>
</body>
<script>
  Var app=angular.module("myApp",[]);
  app.controller('myController', function($scope){
      $scope.name="Hello Geeks!";
      $scope.check=" ";
      $scope.rad=" ";
      $scope.num=" ";
      $scope.mail=" ";
      $scope.url=" ";
      $scope.date1=" ";
      $scope.date2=" ";
      $scope.time1=" ";
      $scope.mon=" ";
      $scope.we=" ";
      $scope.choice=" ";
      $scope.c=function(){
      $scope.choice=true;
    };
  });
</script>
</html>
```

Output :

Input Box -

Name - [Hello Geeks!]

Hello Geeks!

check box - ☑

true

Radio box - ⊙

Number - [12454512]

12454512

Email: example@123.com

example@123.com

url - https://www.geeksforgeeks.org

https://www.geeksforgeeks.org

Date: 28-04-2023

Todays date: Thu Apr 28 2023 00:00:00 GMT+0530 (Indian standard Time)

Datetime-local - 28-04-2023 12:00 AM

Thu Apr 28 2023 00:00:00 GMT+0530 (Indian standard Time),

Time - 12:00 AM

Fri Apr 29 2023 00:00:00 GMT+0530 (Indian standard Time)'

Month - April 2023

Tue Apr 29 2023 00:00:00 GMT+0530 (India standard Time)'

week - week 09, 2023

Thu Feb 28 2023 00:00:00 GMT+0530 (India standard Time)'

c) Using angular JS fetching data from MySQL

Source code:

index.php

```
<html>
  <head>
    <title> Angular JS Tutorial with PHP</title>
    <link rel = "style sheet".
        href = "https://maxcdn.bootstrapcdn.com/
        bootstrap/3.3.6/css/bootstrap.min.css"/>
```

```
<script src="http://ajax.googleapis.com/
  ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<head>
<body style="background:yellow">
  <div class="container" style="width:
                                500px";>
    <h3 align="center"> Angular JS Tutorial
                                </h3>
    <div ng-app="my app" ng-controller=
          "user controller" ng-init="display
                                Data()">
      <label> Country Code </label>
      <input type="text" name="" ng-model=
          "id-send" class="form-control"/>
      <label> Country </label>
      <input type="text" name="" ng-model
        ="name-send" class="form-control"/>
      <input type="~~btn-info~~" "submit" name=
        "btn Insert" class="btn btn-info"
        ng-click="insertData()" value="ADD"/>
      <table class="table table-bordered"
          style="background:# edebee">
          <tr>
          <th> Code </th>
          <th> Country </th>
          </tr>
          <tr ng-repeat="x in codes">
      <td> {{x.id}} </td>
      <td> {{x.name}} </td>
```

```
</tr>
</table>
</div>
</div>
</body>
</html>
<script>
var app= angular.module('myapp',[]);
app.controller('usecontroller', function($scope,
                                $http) {
    $scope.insertData = function(){
$http.post("insert.php",{'id_ajax':
    $scope.id_send,'name_ajax':$scope.
    name_send}).success(function(data)
    { alert(data);
    $scope.name_send=null;
    $scope.id_send=null;
    $scope.displayData();
});

    $scope.displayData = function(){
    $http.get("select.php").success(function
                                (data){
    $scope.codes=data;
    });
});
</script>
```

insert.php:

```
<? php
    $connect= mysqli_connect("localhost","root",
                            "","test");
```

```
$data = json_decode(file_get_contents("php://
if (count($data)>0){          input"));
    $id_received = mysqli_real_escape_string
        c$connect, $data->id_ajax);
    $ name_received =mysqli_real_escape_
        string c$connect, $data->name_
                                ajax);
    $ query = "INSERT INTO country (id,name)
                VALUES ('$id_received',
                        '$name_received')";
    if(mysqli_query ($connect,$query)){
        echo "Data Inserted...";
    }
    else {
      echo "Error";
    }
}
?>
```

Select.php:

```
<?php
    $connect= mysqli_connect( "localhost","root","",
                                "test");
    $ output = array();
    $ query = "SELECT id , name FROM country";
    $ result = mysqli_query ($connect , $query);
    if (mysqli_num_rows ($ result)>0) {
        while ($row = mysqli_fetch_array ($result))
          { $output[] = $row;
          }
```

echo json_encode ($output);

}

?>

← → localhost /php/project 1130/index.php

Angular JS Tutorial

Country Code

Country

ADD

code Country

https://localhost 8000/

Angular JS Tutorial

Country Code

Country

ADD

| Code | Country |
|---|---|
| 81 | china |
| 91 | India |
| 21 | Korea |
| 32 | chile |

Exercise: 7

Aim: To write a program to implement AngularJS scope

Description: The scope is the binding part b/w the HTML (view) and the (controller) Javascript. The scope is available for both the view and the controller. An angularJS application consists of:

→ View; which is the HTML

→ Model; which is the data available for the current view.

→ Controller; which is JS function that makes/changes/removes/controls the data.

Source code:

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app="myApp" ng-controller="myctrl">
<input ng-model="name">
<h1> My name is {{name}} </h1>
</div>
<script>
    var app = angular.module('myApp', []);
    app.controller('myctrl', function($scope) {
        $scope.name = "pharani";
    });
</script>
<p> when you change the name in the input field,
```

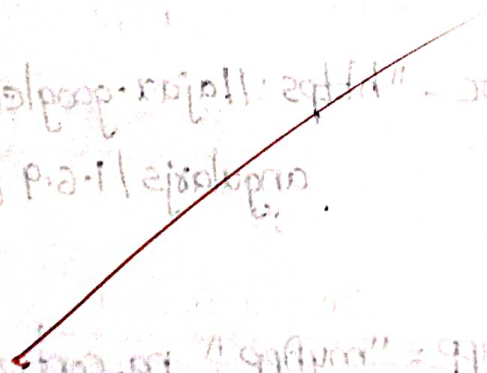
the changes will affect the model, and it will also affect the name property in the controller. </p>
</body>
</html>

Output:

Dharani

My name is Dharani

when you change the name in the input field, the changes will affect the model, and it will also affect the name property in the controller.



Exercise: 8

Aim: To write a program to implement the following using Angular JS

1) Input Validation.

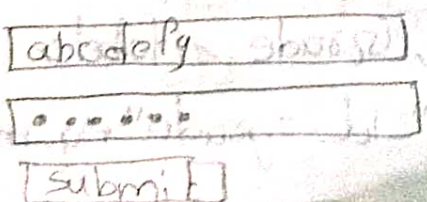
Description: In the program, we will validate the input. Username and password are taken as input from the user. form is submitted as soon as you hit the submit button. It validates the form and displays the message like password is required, Username is required if you leave the field blank.

Source code:

```
<!DOCTYPE html>
<html lang="en">
<body ng-app="myApp">
    <script src="https://ajax.googleapis.com/
    ajax/libs/angularjs/1.8.2/angular.min.js">
                                    </script>
    <div ng-controller="myController">
        <form name="myform" ng-submit=
            "submitform" novalidate>
            <input type="text" ng-model=
            "username" name="username"
            placeholder="Username" required/>
    <br>
    <p ng-show='myform.username.$invalid &&
        myform.username.$touched'> Username is
        required </p>
```

```
<input type="password" ng-model="pass-
-word" name="password" placeholder="Pass-
-word required>
</br>
<p ng-show="myform.password.$invalid
&& myform.password.$touched"> Password
is required </p>
<button type="submit"> Submit </button>
</form>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myController', function($scope)
{ $scope.submitform = function() {
if ($scope.myform.$valid) {
alert('Form submitted!');
} else {
$scope.myform.username.$setTouched
();
$scope.myform.password.$setTouched();
};
});
</script>
</body>
</html>
```

Output:



Exercise: 10.

Aim: To write a program to create an application for student Record using AngularJs.

Description:

In this program, we will create an application, where we can fill the student form containing input fields like Name, Age, Grade, email, ... On submitting the form, the student Record will be displayed in the table.

Source code:

```
<html ng-app = "myApp">
<head>
    <title> Student Form </title>
    <script src = "https://ajax.googleapis.com/
    ajax/libs/angularjs/1.8.2/angular.min.js">
</head>                                    </script>
<body ng-controller = "studentController">
    <h2> Student Form </h2>
    <form ng-submit = "submitForm()">
        <label for = "name"> Name: </label>
        <input type = "text" id = "name" ng-model =
        "student.name" required> <br/>
        <label for = "age"> Age </label>
        <input type = "number" id = "age" ng-model =
        "student age" required> <br/>
        <label for = "grade", Grade </label>
        <input type = "text" id = "grade" ng-model =
```

```
"student.email" required><br/>
<label for = address> Address </label>
<input type = "text" id = "address" ng-model=
    "student.address" required><br/>
<label for = "phone"> Phone: </label>
<input type = "tel" id = "phone" ng-model=
    "student.phone" required> <br/>
<button type = "submit"> Submit </button>
<button type = "button" ng-click = "resetForm()">
                Reset </button>

</form>
<br/>
<br/>
<table border = 2>
<h2> student Records </h2>
<tr>
    <th> Name </th>
    <th> Age </th>
    <th> Grade </th>
    <th> Email </th>
    <th> Address </th>
    <th> Phone </th>
</tr>
<tr ng-repeat = "record in studentRecords">
    <td> {{ record.name }} </td>
    <td> {{ record.age }} </td>
    <td> {{ record.grade }} </td>
    <td> {{ record.email }} </td>
    <td> {{ record.address }} </td>
```

```
    <td> {{ record.phone }} </td>
  </tr>
</table>
<script>
  angular.module('myApp', [])
      .controller('StudentController', ['$scope',
                                function($scope) {
    $scope.student = { };
    $scope.studentRecord = [];
    $scope.submitForm = function() {
      $scope.studentRecords.push(angular.copy(
                        $scope.student));
      $scope.resetForm();
    };
    $scope.resetForm = function() {
        $scope.student = { };
    };
  }]);
</script>
</body>
</html>
```

Output:

Student Form

Name:

Age:

Grade:

Email:

Address:

Phone:

Submit | Reset

Student Records.

| Name | Age | Grade | Email | Address | phone |
|---|---|---|---|---|---|
| abcde | 20 | 3 | abcdeoo2@gmail.com | PQRST | 2345678910 |

Exercise: 12

Aim: Design and demonstrate invoking data using Jscript from MongoDB.

Description:

MongoDB is a document-oriented No SQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consists of key-value pairs which are the basic unit of data in MongoDB.

Create Collection:

```
const MongoClient = require("mongodb");
const url = "mongodb://localhost:27017/";
MongoClient.connect(url, function(err, db){
    if(err) throw err;
    var dbo = db.db("mydb");
    dbo.createCollection("customer", function
                                    (err, res){
        if(err) throw err;
        console.log("collection created");
        db.close();
    });
});
```

Output:

collection created

Insert into Collection:

```
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");
      Var myobj = [
    {name:'John', address:'Highway 71'},
    {name:'Peter', address:'lawstreet.4'},
    {name:'Amy', address:'Apple st 652'},
    {name:'Hannah', addres:'Mountain 21'},
    {name:'Michael', address:'valley 345'},
  ];
  dbo.collection("customers").insertMany(
      myobj, function(err, res){
        if(err) throw err;
        console.log("Number of documents
                inserted:"+ res.insertedCount);
        db.close();
      });
  });
```

Actual output:

| Number of documents inserted:5 |
|---|

Final document based on fields in collection:

```
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.collection("customers").find({},
   { projection: {_id:0, name:1, address:1}
  }).toArray(function(err, result) {
        if(err) throw err;
        console.log(result);
        db.close();
    });
});
```

Actual Output:

```
{name: 'John', address: 'Highway 71'},
{name: 'peter', address: 'Lowstreet 4'},
{name: 'Amy', address: 'Apple st 652'},
{name: 'Michael', address: 'Valley 345'},
{name: 'John', address: 'Highway 71'},
{name: 'peter', address: 'lowstreet 4'},
{name: 'Amy', address: 'Apply st 652'},
{name: 'Hannah', address: 'Mountain 21'},
{name: 'Michael', address: 'Valley 345'}
```

Delete documents:

```
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");
    var myquery = { address: 'Mountain4' };
    dbo.collection("customer").deleteOne(
        myquery, function(err, obj){
        if(err) throw err;
        console.log("1 document deleted");
        db.close();
    });
});
```

Actual output:

```
c:\Program Files\nodejs\mongodb>node deletenode.js
1 Document deleted
```