NODEJS

by Harsha Vardhan (UI Expert)

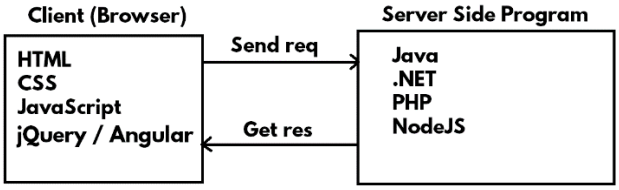
Contents

NodeJS

NODEJS

Introduction to NodeJS

- NodeJS is a "Server side platform", which is used to create "server side programs" in web applications.
- The server side program receives request from browser, process the request and sends response to the browser. It connects to database & maintains security.



- NodeJS was released in 2009 and the latest version is "9".
- NodeJS was developed by "NodeJS foundation"; leader is "Ryan Dahl".

Features and Advantages of NodeJS

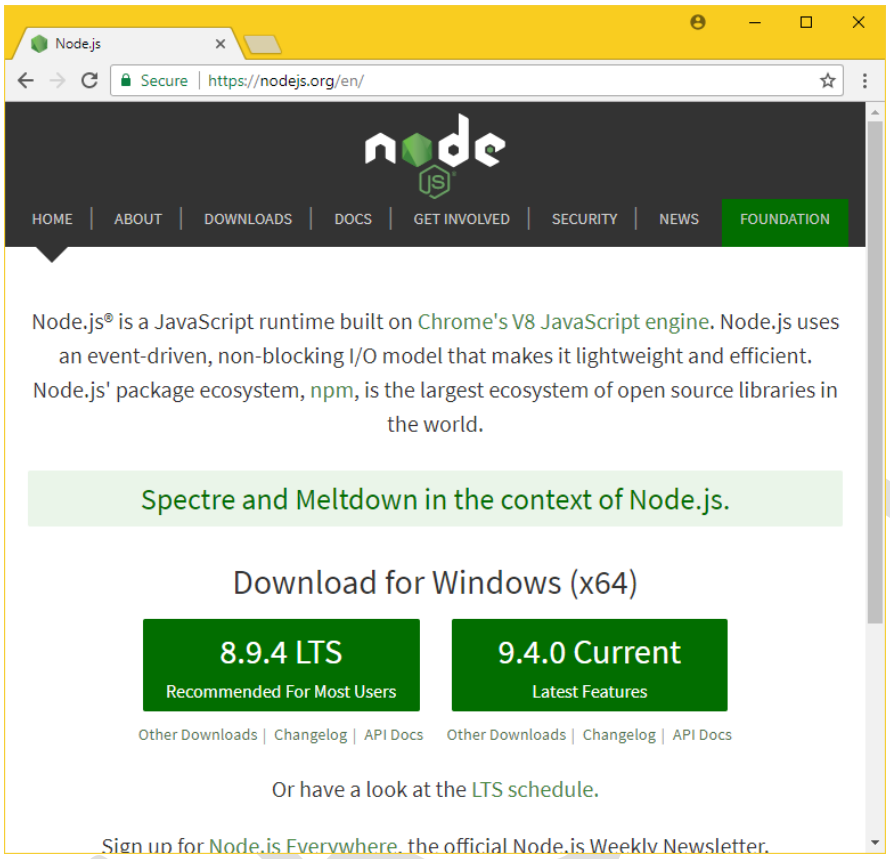
- NodeJS is very light-weight and very fast.
- NodeJS is JavaScript based and easy.
- Non-Blocking I/O Model: The file and database operations are done asynchronously. That means the server executes another request while a request's file connection / database connection is under progress.
- NodeJS internally uses Google's V8 JavaScript Engine. It is the fastest JavaScript Engine.
- NodeJS is free-to-use.
- NodeJS is open source.
- NodeJS is Cross-platform. It supports Windows, LINUX and Mac.

Steps to Prepare First Example in NodeJS

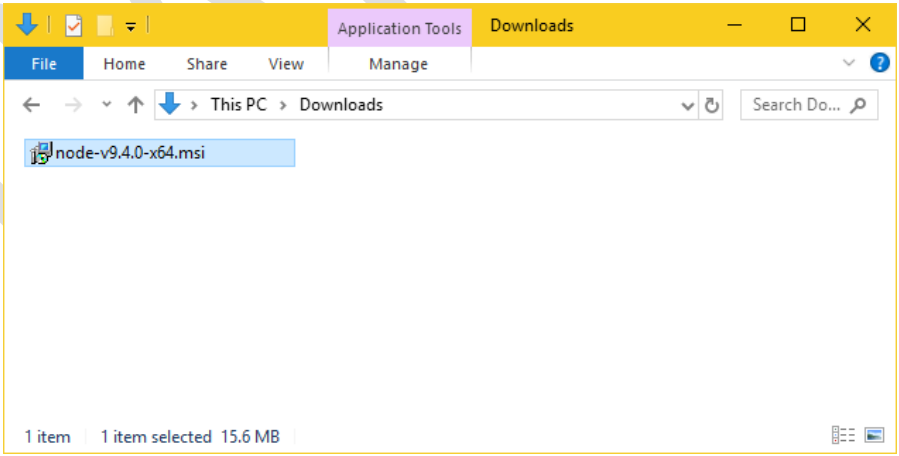
1) Installing NodeJS
2) Create Application Folder
3) Installing "Visual Studio Code"
4) Create NodeJS Program
5) Execute the NodeJS Program

1) Installing NodeJS

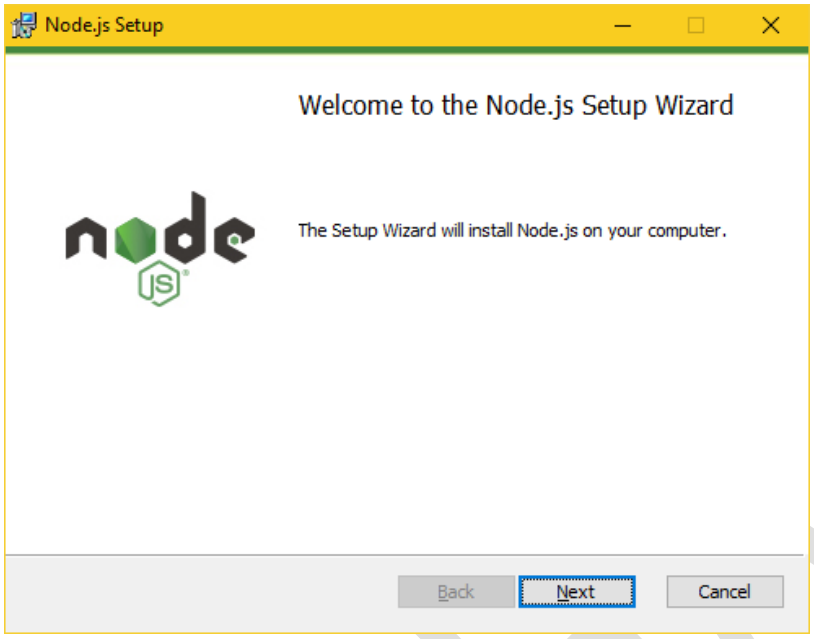
- Go to "https://nodejs.org".



- Click on "9.4.0 Current". **Note:** The version number can be very. Choose the latest version.
- You will download a file called "node-v9.4.0-x64.msi".

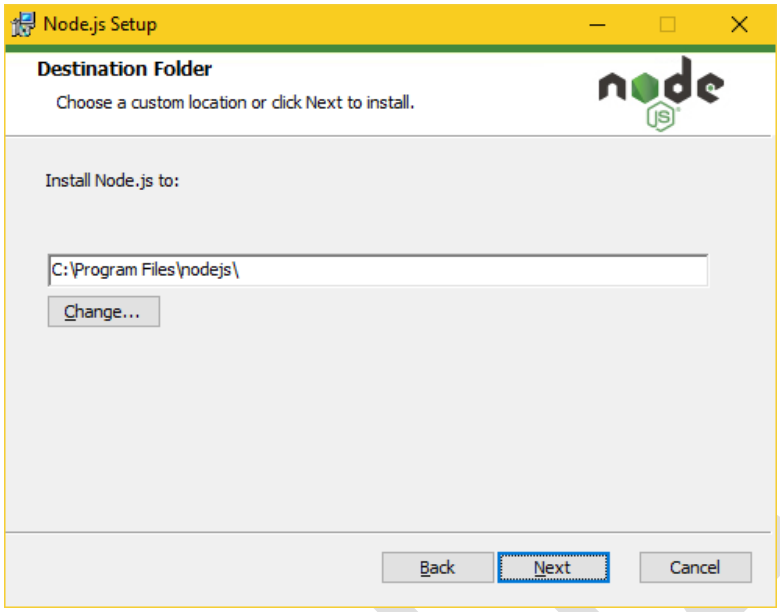


- Go to Downloads folder and double click on "node-v9.4.0-x64.msi".

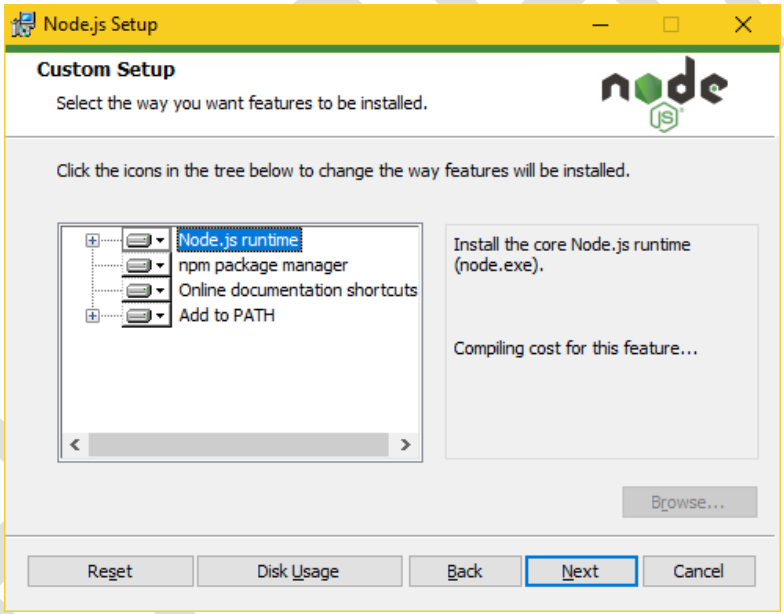


- Click on "Next".

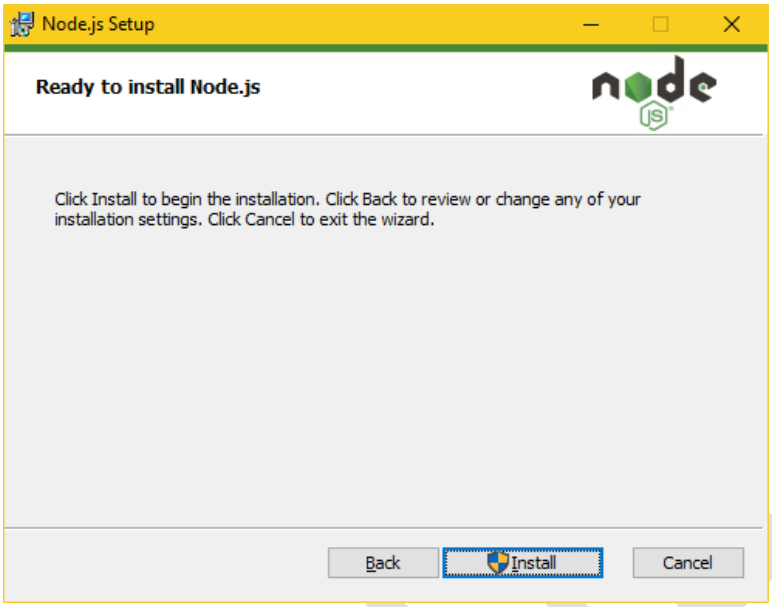
- Check the checkbox "I accept the terms in the License Agreement" and click on "Next".



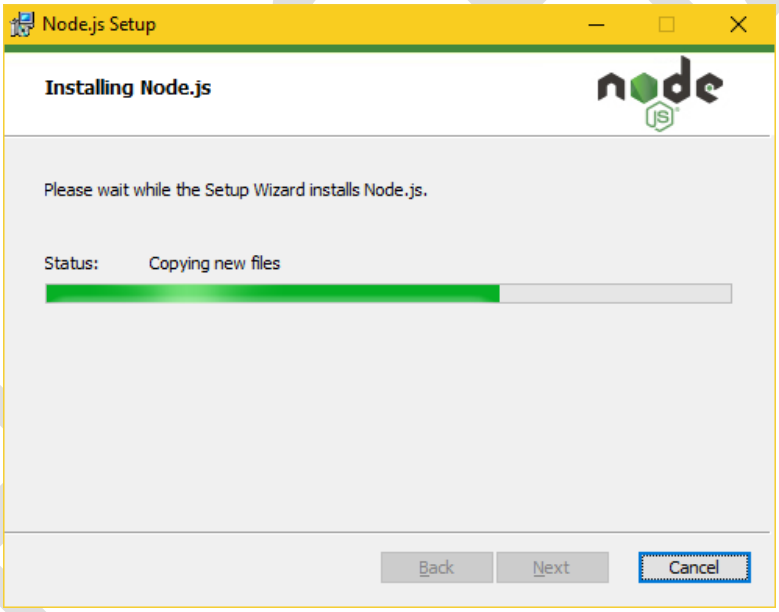
- Click on "Next".



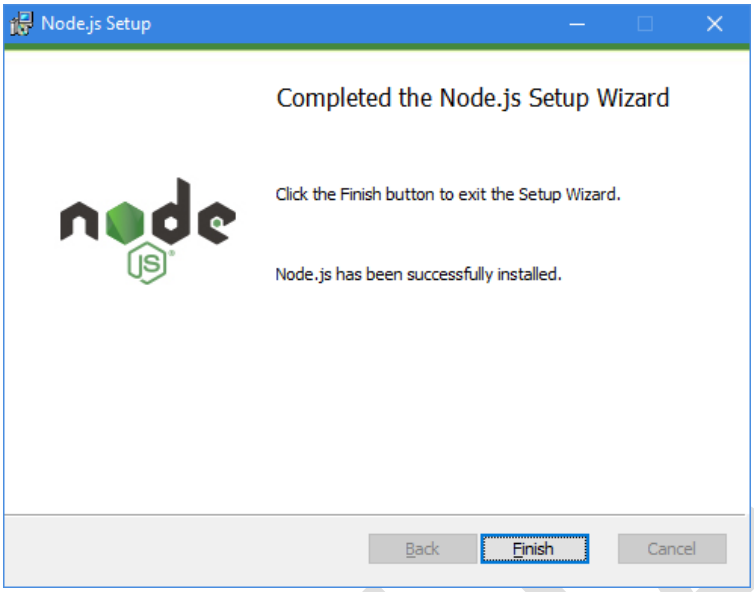
- Click on "Next".



- Click on "Install".
- Click on "Yes".



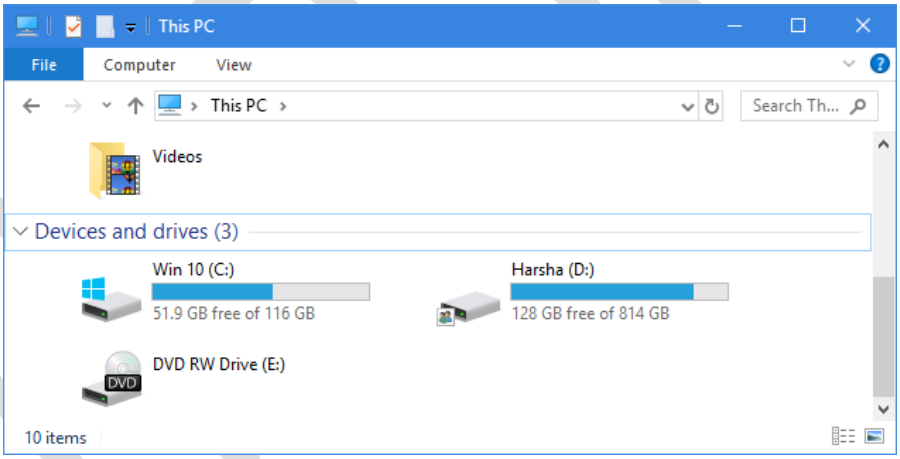
- Installation is in progress now.



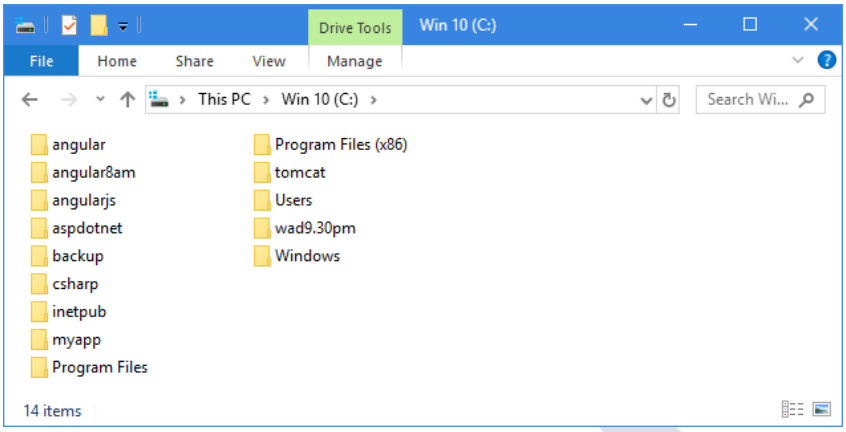
- After installation is completed, click on "Finish". Now NodeJS installation is finished.
- **Note:** Automatically it adds the nodejs installation directory in the "Path" at environment variables of your computer.

2) Create Application Folder

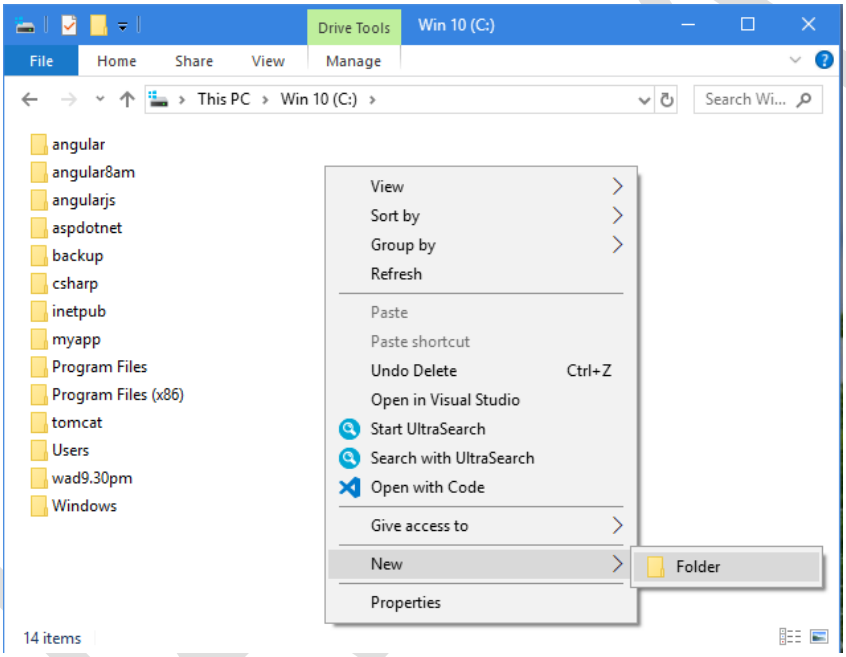
- Go to "Computer" (or) "This PC".



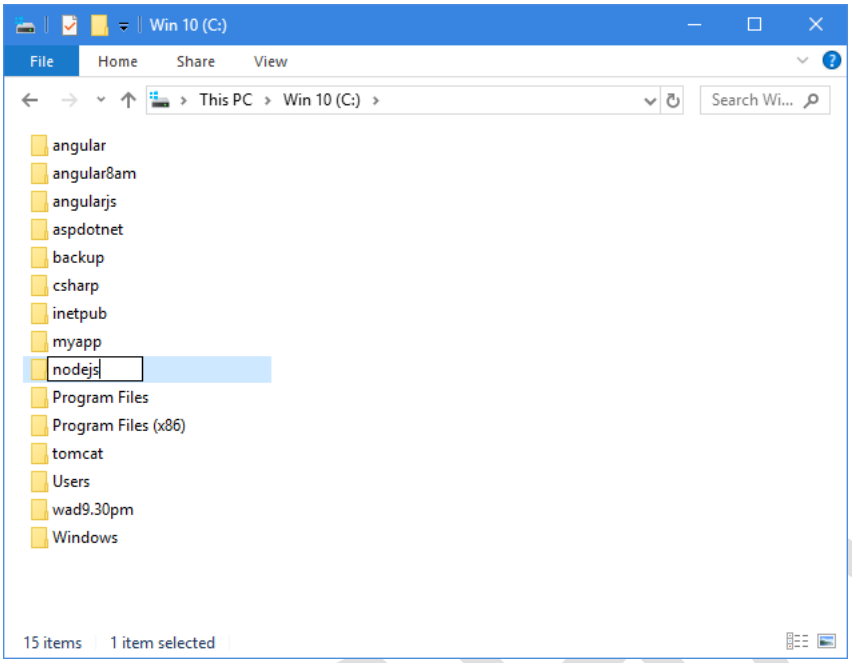
- Go to "c:\".



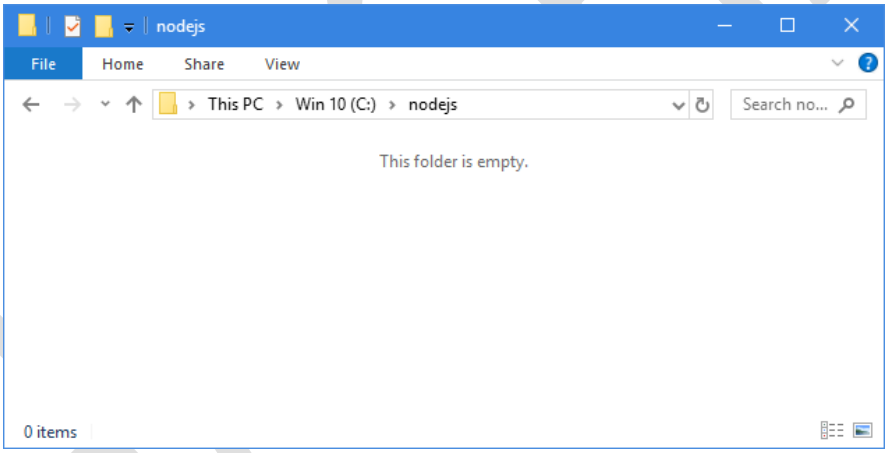
- Right click on the empty area and click on "New Folder".



- Type the folder name as "nodejs" and press Enter.

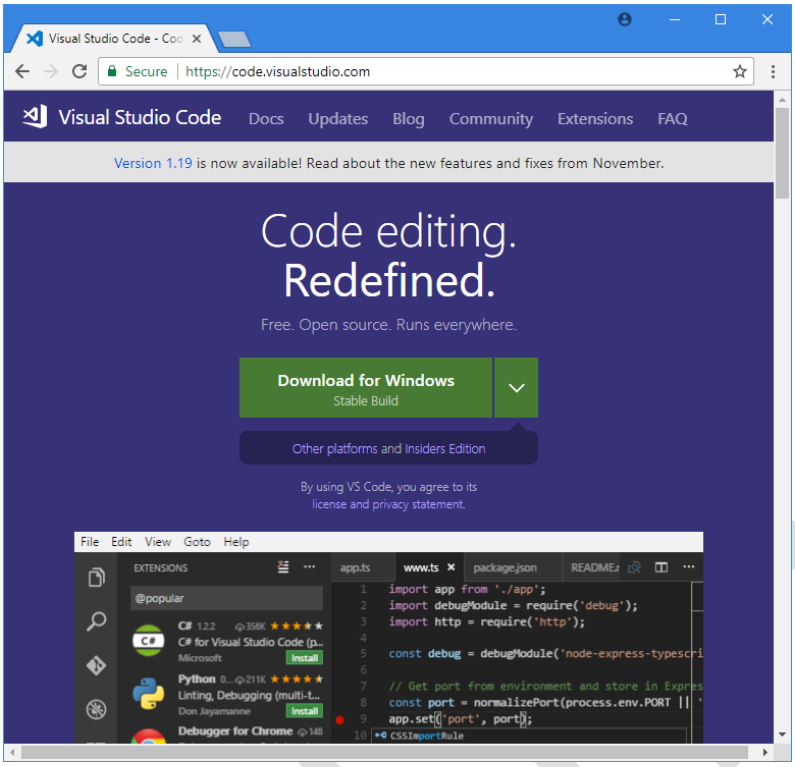


- The "c:\nodejs" folder is ready now.

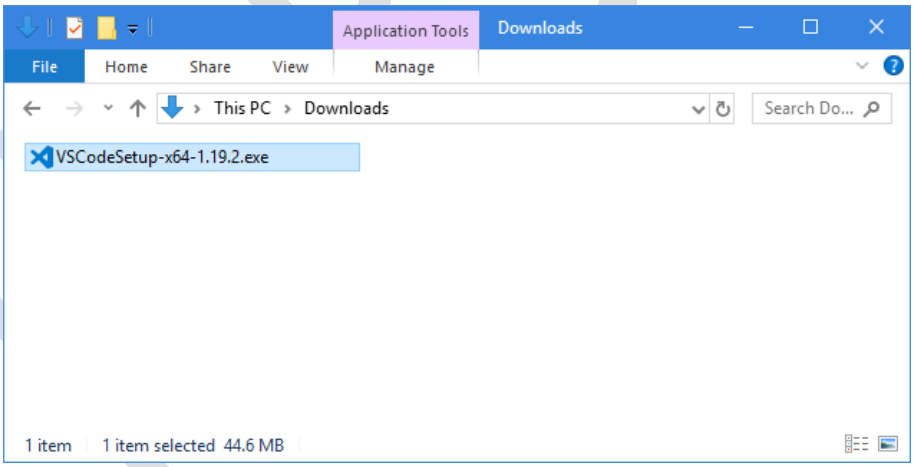


3) Installing Visual Studio Code

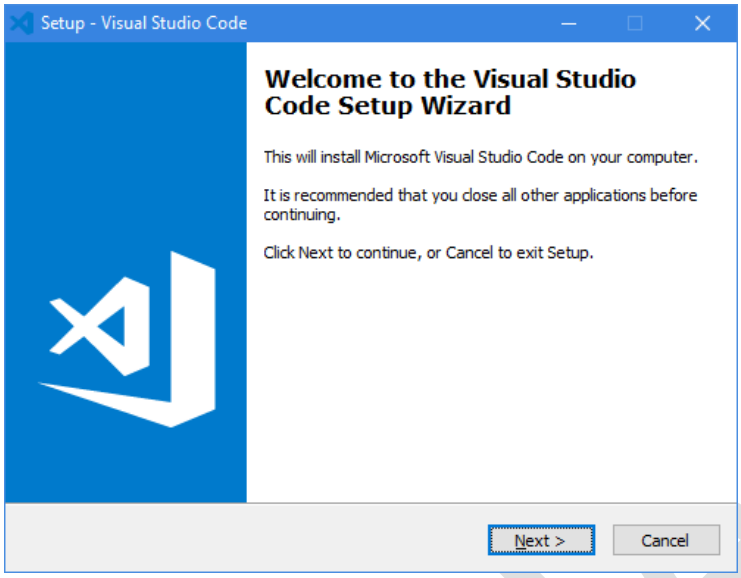
- "Visual Studio Code" is the recommended editor for nodejs.
- Go to https://code.visualstudio.com



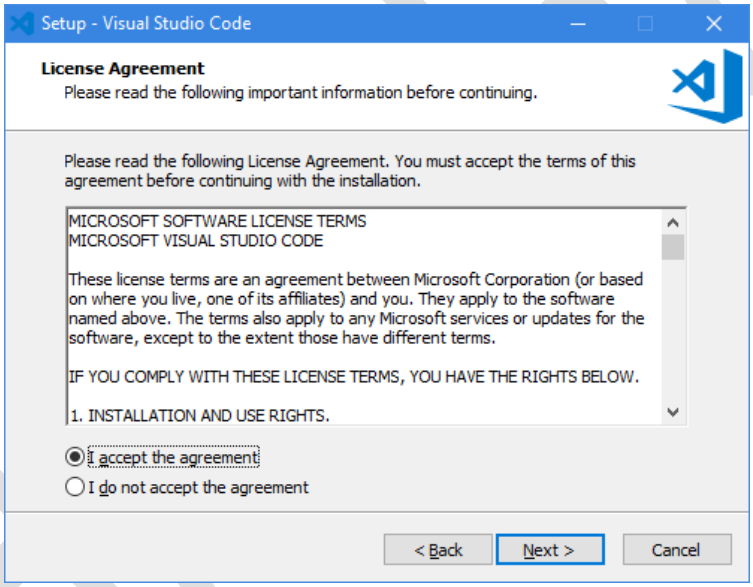
- Click on "Download for Windows".
- **Note:** The version number may be different at your practice time.
- Go to "Downloads" folder; you can find "VSCodeSetup-x64-1.19.2.exe" file.



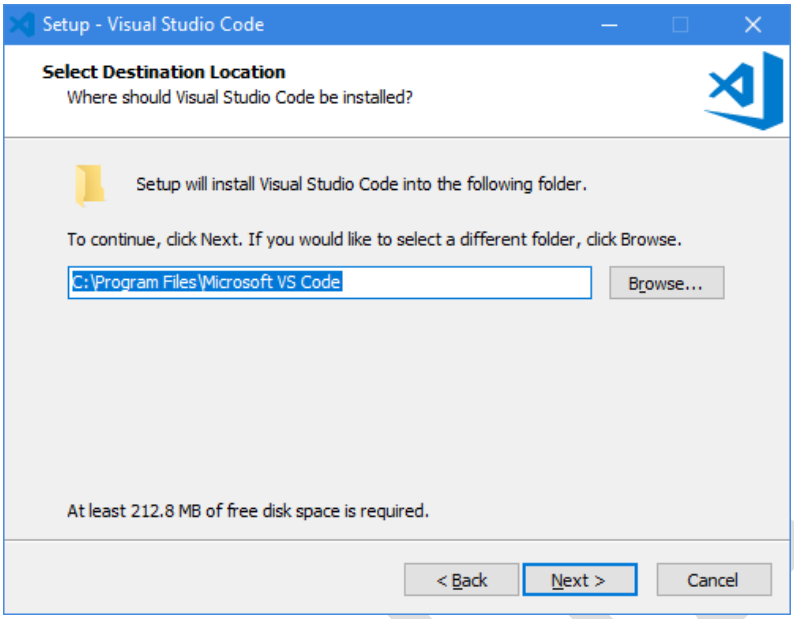
- Double click on "VSCodeSetup-x64-1.19.2.exe" file.
- Click on "Yes".



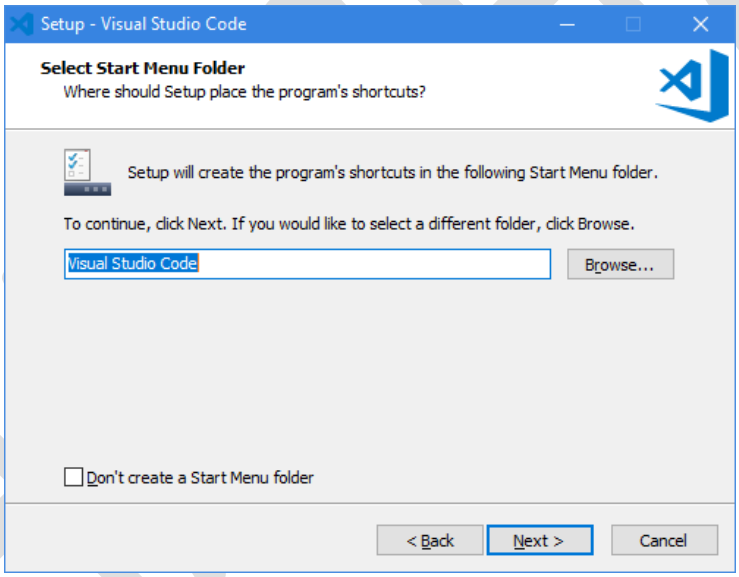
- Click on "Next".



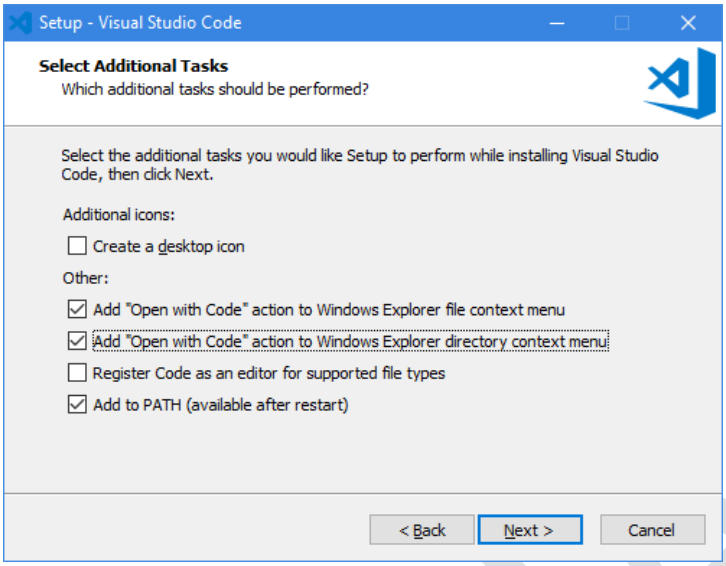
- Click on "I accept the agreement".
- Click on "Next".



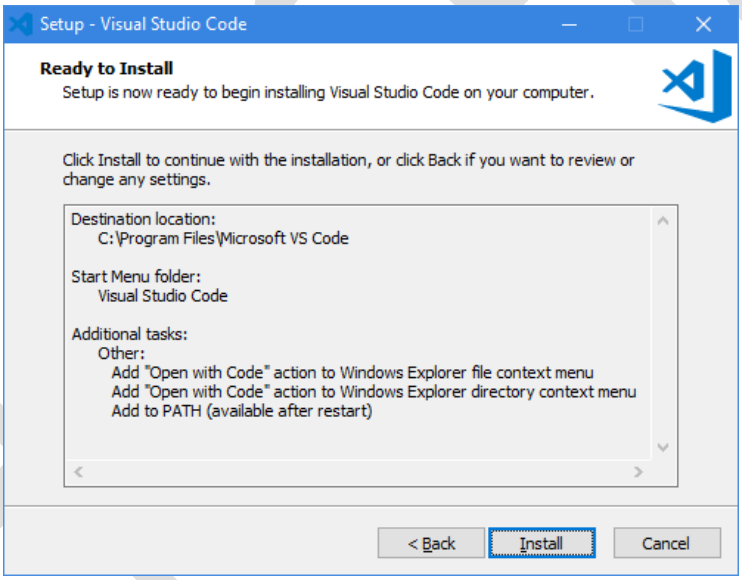
- Click on "Next".



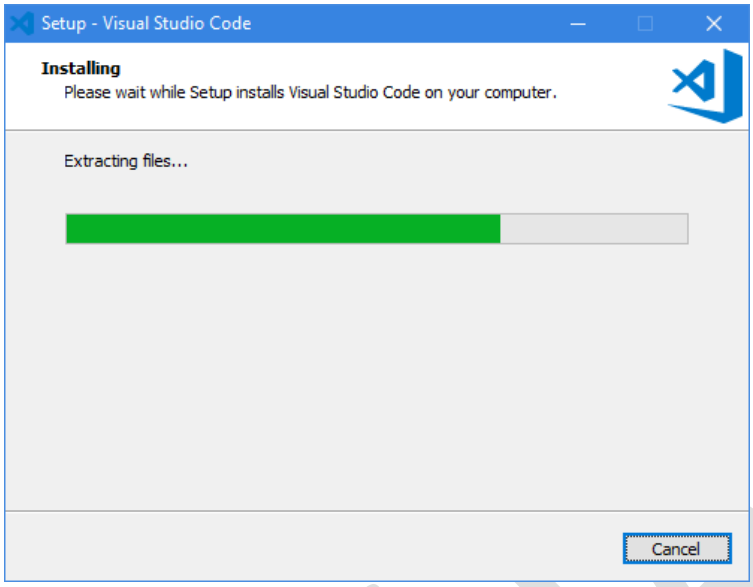
- Click on "Next".



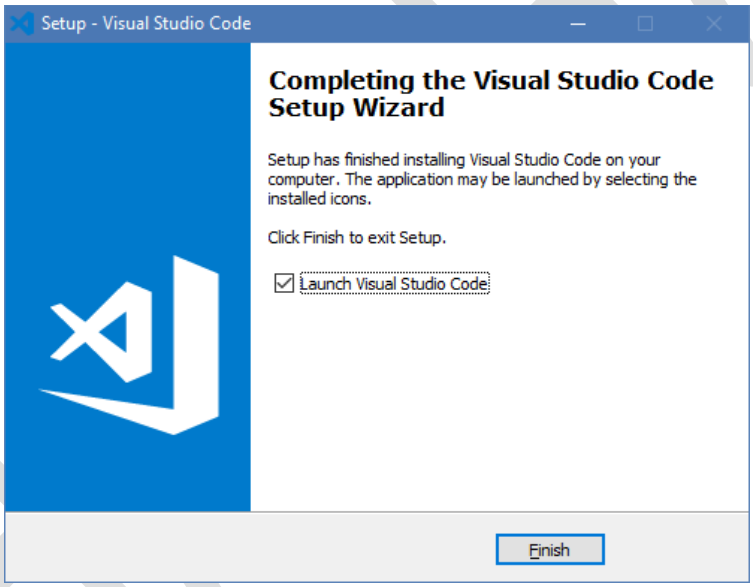
- Check the checkbox "Add Open with Code action to Windows Explorer file context menu".
- Check the checkbox "Add Open with Code action to Windows Explorer directory context menu".
- Click on "Next".



- Click on "Install".



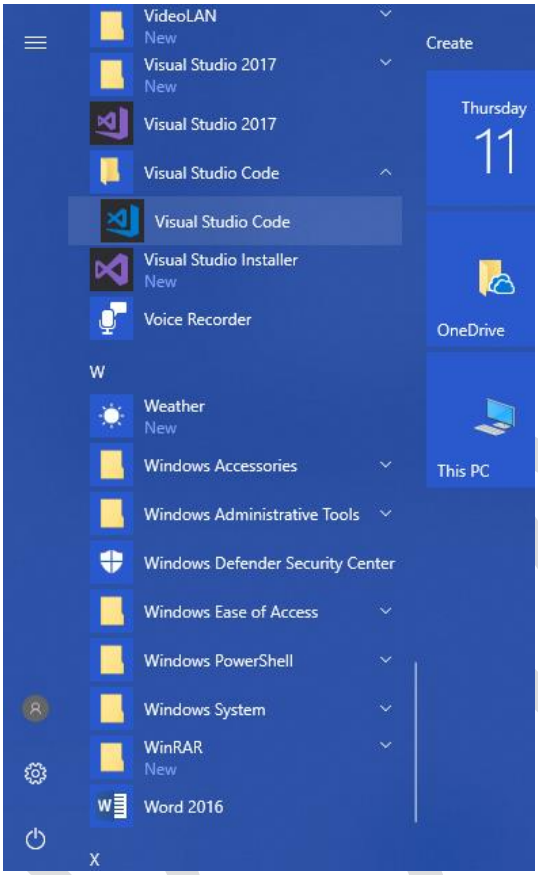
- Installation is going on....

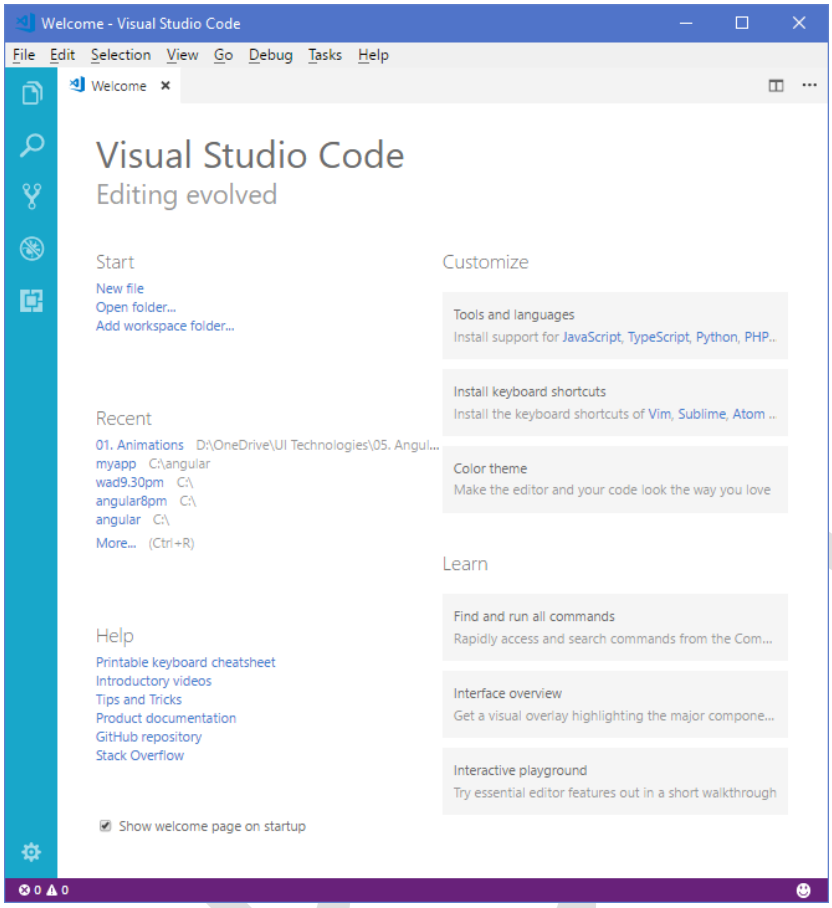


- Click on "Finish".

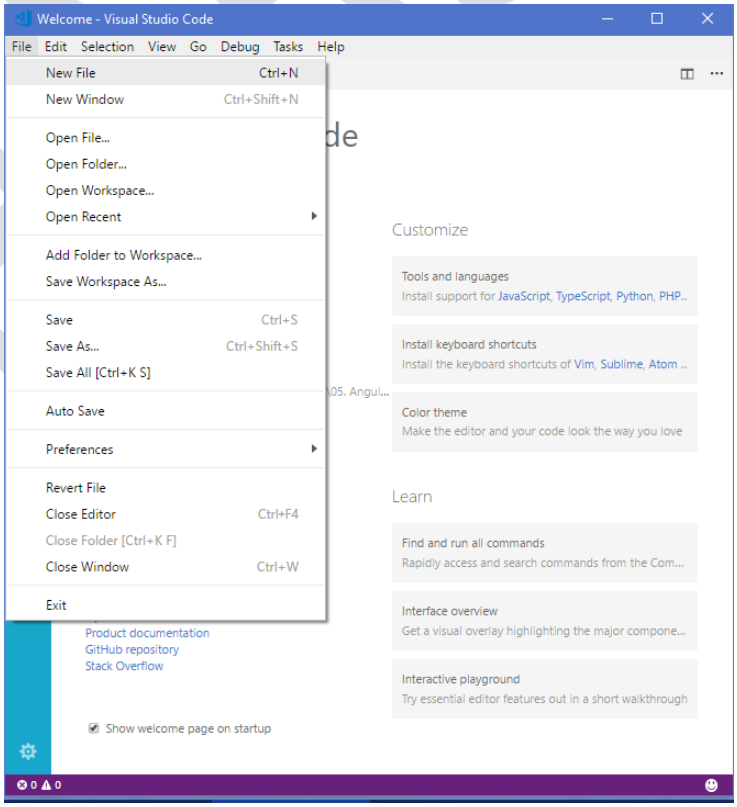
4) Create NodeJS Program

- Open "Visual Studio Code", by clicking on "Start" – "Visual Studio Code" – "Visual Studio Code".

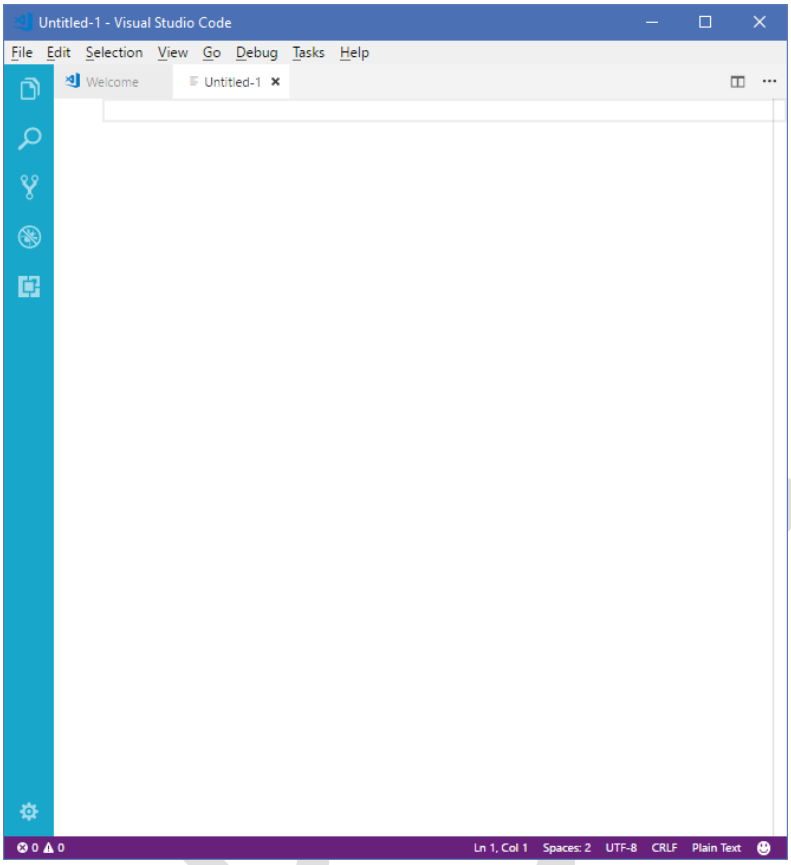

VideoLAN
New
Visual Studio 2017
New
Visual Studio 2017
Visual Studio Code
Visual Studio Code
Visual Studio Installer
New
Voice Recorder
W
Weather
New
Windows Accessories
Windows Administrative Tools
Windows Defender Security Center
Windows Ease of Access
Windows PowerShell
Windows System
WinRAR
New
Word 2016
X
Create
Thursday
11
OneDrive
This PC



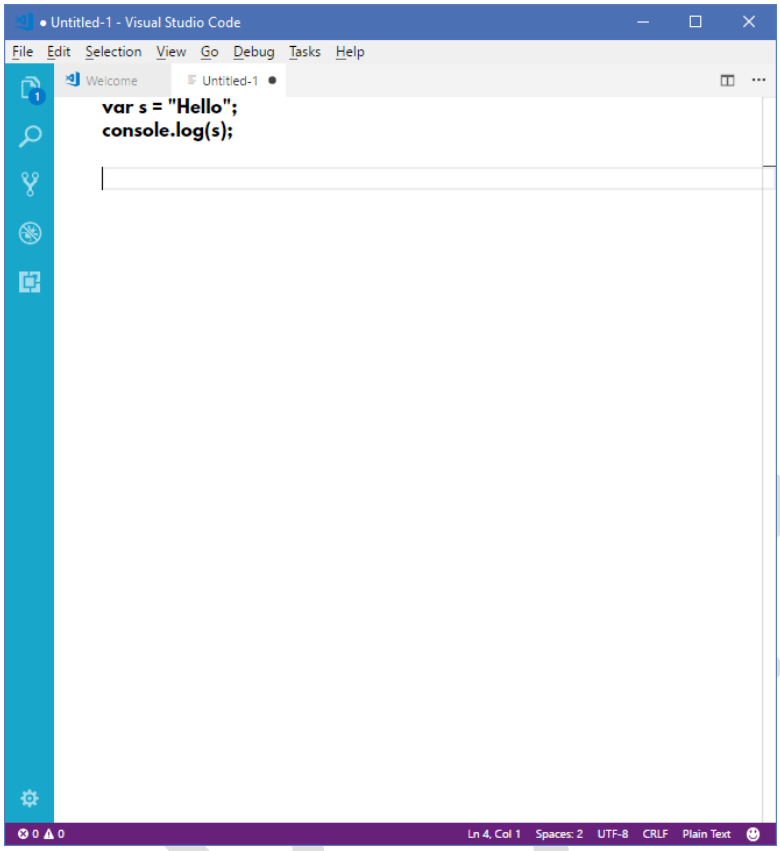
- Visual Studio Code opened.



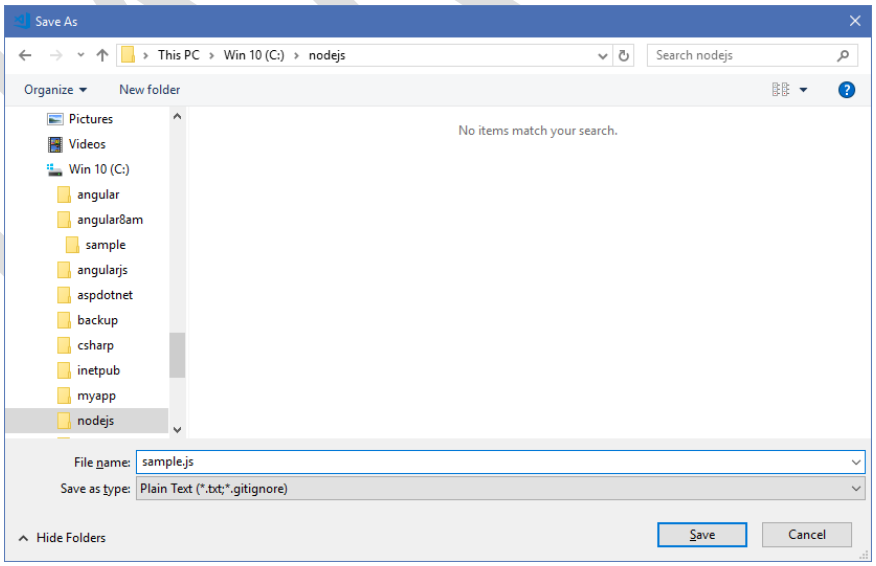
- Go to "File" – "New File".



- Type the program as follows:



- Go to "File" menu – "Save" (or) Press Ctrl+S.



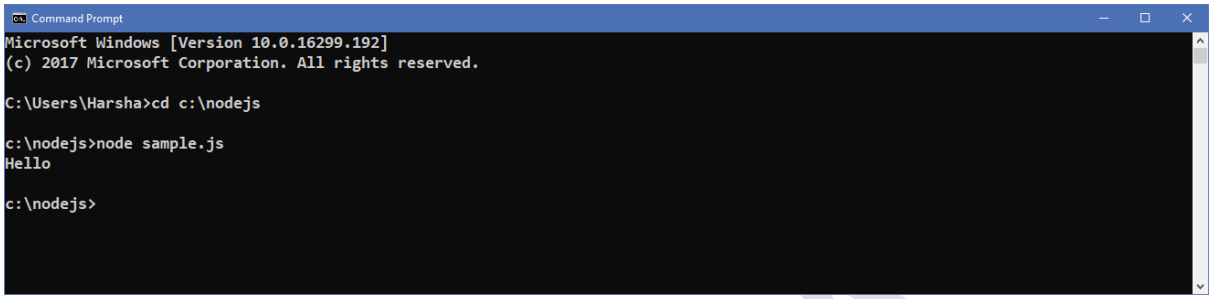
- Select "c:\nodejs" folder and enter the filename as "sample.js".
- Click on "Save".
- Now the nodejs file (c:\nodejs\sample.js) is ready.

5) Execute the NodeJS Program

- Open Command Prompt and enter the following command:

  cd c:\nodejs

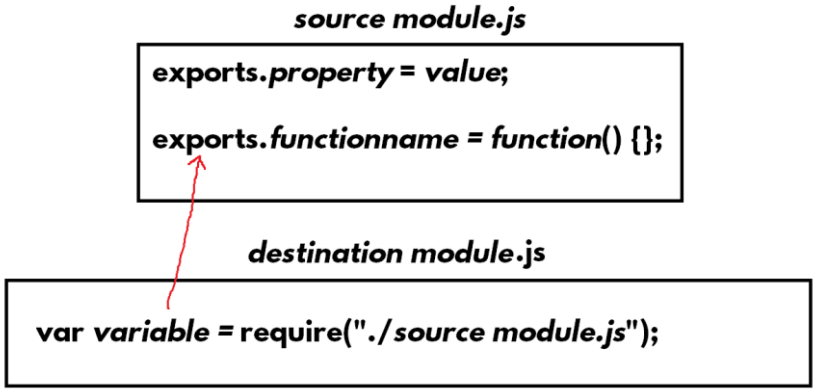
  node sample.js



- **Output:** Hello

Modules

- Module is a "JavaScript file" (filename.js).
- Module can share its properties and functions to other module. The source module exposes an object called "exports", which contains all the properties and functions to be shared. The destination module acquires (gets) the "exports" object of the source module.



- "./" refers to "current folder".

Modules - Example

c:\nodejs\sourcemodule.js

```
var msg = "Hello";

function fun1(a, b)
{
    var c = a + b;
    return (c);
};

function fun2(a, b)
{
        var c = a - b;
        return (c);
}
```

```
exports.message = msg;
exports.add = fun1;
exports.subtract = fun2;
```

c:\nodejs\destinationmodule.js

```
var data = require("./sourcemodule.js");

console.log(data.message);
console.log("Sum: " + data.add(10, 3));
console.log("Difference: " + data.subtract(10, 3));
```

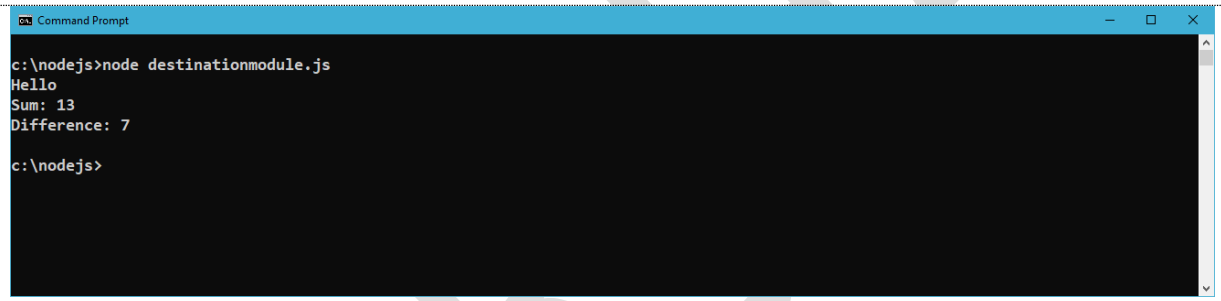
Execution

- Open Command Prompt

  cd c:\nodejs

  node destinationmodule.js

Output:



"fs" module

- The "fs" module provides a set of functions to manipulate files and folders in the harddisk.
- **Important functions of "fs" module:**
  - readFile()
  - readFileSync()
  - writeFile()
  - writeFileSync()
  - rename()
  - unlink()
  - stat()
  - readdir()

readFile()

- This function is used to read content of an existing file, asynchronously.
- **Steps:**
  - **Acquire "fs" module:**

    var fs = require("fs");
  - **Read content of the file:**

    fs.readFile("filename", "encoding", callbackfunction);

```
function callbackfunction(error, data)
{
    //error = Represents exception details
    //data = Represents file content
}
```

encoding = utf8 / binary

The callback function will be called automatically, after reading the file content.

readFile() - Example

c:\nodejs\file.txt

```
Hello
How
are you
```

c:\nodejs\readfile.js

```
var fs = require("fs");
fs.readFile("file.txt", "utf8", fun1);
console.log("After readFile");

function fun1(error, data)
{
    if (error)
    {
        console.log(error);
    }
    if (data)
    {
        console.log(data);
    }
}
```

Execution

- Open Command Prompt

```
cd c:\nodejs
node readfile.js
```

Output:

readFileSync()

- This function sis used to read content of an existing file, synchronously.
- **Steps:**
  - **Acquire "fs" module:**

    var fs = require("fs");
  - **Read content of the file:**

    var data = fs.readFileSync("filename", "encoding");

encoding = utf8 / binary

The "data" variable (name can be anything) represents the file content.

readFileSync() - Example

c:\nodejs\file.txt

```
Hello
How
are you
```

c:\nodejs\readfilesync.js

```
var fs = require("fs");
console.log("Before readFile");

try
{
    var data = fs.readFileSync("file.txt", "utf8");
    console.log(data);
}
catch (error)
{
    console.log(error);
}
console.log("After readFile");
```

Execution

- Open Command Prompt

  cd c:\nodejs

  node readfilesync.js

Output:

writeFile()

- This function is used to write content into a file, asynchronously.
- **Steps:**
  - **Acquire "fs" module:**

    var fs = require("fs");
  - **Write content into the file:**

```
fs.writeFile("filename", "filecontent", "encoding", callbackfunction);
function callbackfunction(error)
{
    //error = Represents exception details
}
```

encoding = utf8 / binary

The callback function will be called automatically, after writing the content to the file.

writeFile() - Example

c:\nodejs\writefile.js

```
var fs = require("fs");
var data = "Hello\nHow\nare you";
fs.writeFile("file.txt", data, "utf8", fun1);
console.log("After writeFile");

function(error)
{
   if (error)
   {
       console.log(error);
   }
   else
   {
       console.log("Written successfully");
   }
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

  node writefile.js

Output:

writeFileSync()

- This function is used to write content to a file, synchronously.
- **Steps:**
  - **Acquire "fs" module:**

    var fs = require("fs");
  - **Write content to the file:**

    var data = fs.writeFileSync("filename", "file content", "encoding");

encoding = utf8 / binary

writeFileSync() - Example

c:\nodejs\writefilesync.js

```
var fs = require("fs");
console.log("Before writeFile");

var data = "Hello\nHow\nare you";
try
{
   fs.writeFileSync("file.txt", data, "utf8");
   console.log("Written successfully");
}
catch (error)
{
   console.log(error);
}
console.log("After writeFile");
```

Execution

- Open Command Prompt

  cd c:\nodejs

  node writefilesync.js

Output:

rename()

- This function is used to rename a file, asynchronously.
- **Steps:**
  - **Acquire "fs" module:**

    var fs = require("fs");

- **Rename the file:**

```
fs.rename("old filename", "new filename", callbackfunction)
function callbackfunction(error)
{
    //error = Represents exception details
}
```

The callback function will be called automatically, after renaming the file.

rename() - Example

c:\nodejs\abc.txt

```
Hello
How
are you
```

c:\nodejs\rename.js

```
var fs = require("fs");

fs.rename("abc.txt", "xyz.txt", fun1);
function fun1(error)
{
    if (error)
    {
        console.log(error);
    }
    else
    {
        console.log("Done");
    }
}
```

Execution

- Open Command Prompt

```
cd c:\nodejs
node rename.js
```

Output:

Copying the file

- There is no any pre-defined function to copy the file; To do so, we can read content of the source file and write the same to destination file.

Copying the file - Example

c:\nodejs\abc.txt

```
Hello
How
are you
```

c:\nodejs\copy.js

```
var fs = require("fs");

fs.readFile("abc.txt", "utf8", fun1);
function fun1(error, data)
{
    if (error)
    {
        console.log(error);
    }
    else
    {
        fs.writeFile("xyz.txt", data, fun2);

        function fun2(error)
        {
            if (error)
            {
                console.log(error);
            }
            else
            {
                console.log("copied");
            }
        }
    }
}
```

Execution

- Open Command Prompt

```
cd c:\nodejs
node copy.js
```

Output:

unlink()

- This function is used to delete the file.
- **Steps:**
  - **Acquire "fs" module:**

    ```
    var fs = require("fs");
    ```

  - **Delete the file**

    ```
    fs.unlink("filename", callbackfunction)
    function callbackfunction(error)
    {
        //error = Represents exception details
    }
    ```

The callback function will be called automatically, after deleting the file.

unlink() - Example

c:\nodejs\abc.txt

```
Hello
How
are you
```

c:\nodejs\unlink.js

```
var fs = require("fs");

fs.unlink("abc.txt", fun1);
function fun1(error)
{
    if (error)
    {
        console.log(error);
    }
    else
    {
        console.log("Deleted");
    }
}
```

Execution

- Open Command Prompt

```
cd c:\nodejs
node unlink.js
```

Output:

stat()

- This function is used to get statistics (details) about the file or folder.
- **Steps:**
  - **Acquire "fs" module:**

    var fs = require("fs");
  - **Delete the file**

    fs.stat("filename", callbackfunction)

    function callbackfunction(error, stats)

    {

    //error = Represents exception details

    //stats = Represents file details

    }

The callback function will be called automatically, after getting the file details.

stat() - Example

c:\nodejs\file.txt

```
Hello
How
are you
```

c:\nodejs\stat.js

```
var fs = require("fs");

fs.stat("file.txt", fun1);
function fun1(error, stats)
{
    if (error)
    {
        console.log(error);
    }
    else
    {
        console.log(stats);
    }
```

}

Execution

- Open Command Prompt

  cd c:\nodejs

  node stat.js

Output:

readdir()

- This function is used to read the list of files / folders in the specified folder.
- **Steps:**
  - **Acquire "fs" module:**

    var fs = require("fs");
  - **Delete the file**

    fs.readdir("folderpath", callbackfunction)

    function callbackfunction(error, files)

    {

    //error = Represents exception details

    //files = Represents an array of files and sub folders

    }

The callback function will be called automatically, after getting the files and folders list.

readdir() - Example

Folder Structure:

c:\nodejs\myfolder

- **folder1**
- **folder2**
- **folder3**
- **a.txt**
- **b.txt**

- c.txt

c:\nodejs\readdir.js

```
var fs = require("fs");

fs.readdir("myfolder", function(error, files)
{
   console.log(files);
   for (var i = 0; i < files.length; i++)
   {
       console.log(files[i] + " - " + fs.statSync("./myfolder/" + files[i]).isFile());
   }
});
```

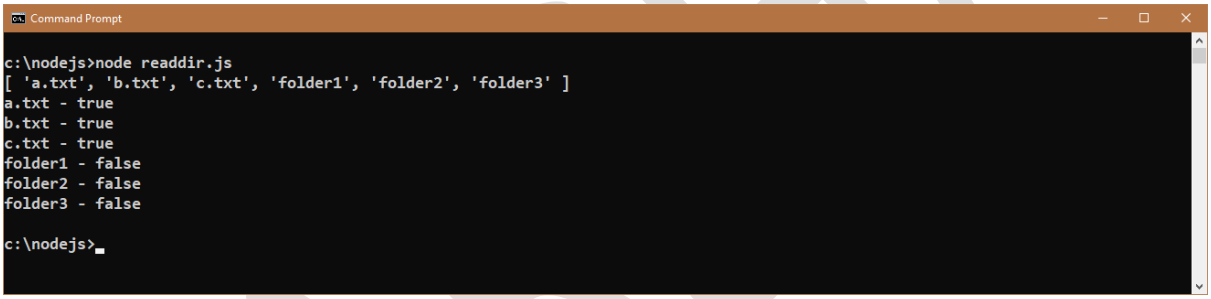
Execution

- Open Command Prompt

  cd c:\nodejs

  node readdir.js

Output:



"events" module

- The "events" module provides a set of functions to handle and raise the events.
- "EventEmitter" is a pre-defined class in "events" module.
- **Steps for working with "events" module**
  - **Acquire "events" module:**

    var events = require("events");
  - **Create an object for "EventEmitter" class:**

    var eventemitter = new events.EventEmitter();
  - **Handle the event / Attach event with a function:**

    eventemitter.on("eventname", callbackfunction);
  - **Raise the event / Call the function:**

    eventemitter.emit("eventname");

Events - Example

c:\nodejs\events.js

```
var events = require("events");
var eventemitter = new events.EventEmitter();
eventemitter.on("mycustomevent", fun1);
```

```
function fun1(arg1, arg2)
{
    console.log("event fired");
    var sum = arg1 + arg2;
    console.log("sum is: " + sum);
}
console.log("event listener added");

setTimeout(fun2, 2000);
function fun2()
{
    eventemitter.emit("mycustomevent", 10, 20);
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

  node events.js

Output:

package.json

- Module is a javascript file (filename.js).
- Package is a collection of modules.
- Package.json file represents meta data (self details about the package).
- **Syntax of "package.json":**

```
{
    "name": "name of the package",
    "version": "version of the package",
    "description": "description of the package",
    "author": "developer name",
    "license": "ISC",
    "repository": "source control url of the application",
    "dependencies":
    {
        "packagename": "version",
        "packagename": "version",
        ...
    }
```

}

- **Installing packages:**

  Command Prompt:

  npm install

Package.json - Example

c:\nodejs\package.json

```
{
 "name": "mypackage",
 "version": "1.0.0",
 "description": "this is my package",
 "author": "Harsha",
 "license": "ISC",
 "repository": "none",
 "scripts":
 {
  "start": "node hello.js"
 }
}
```

c:\nodejs\hello.js

```
console.log("Hello");
```

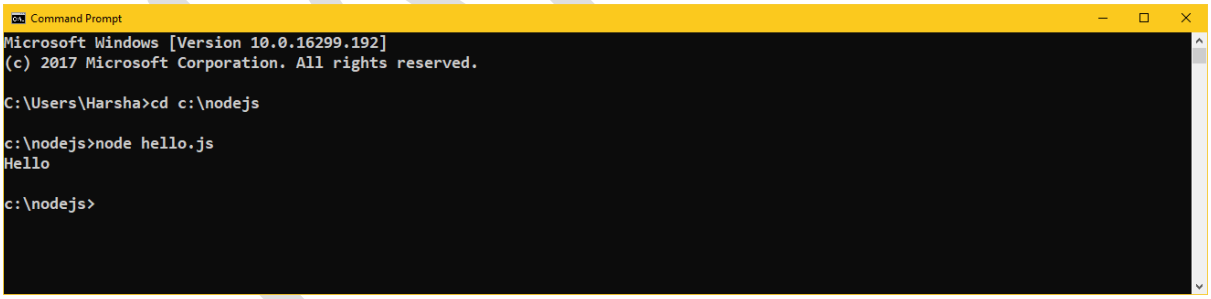
Execution

- Open Command Prompt

  cd c:\nodejs

  node hello.js

Output:



"Mongoose" package

- The "mongoose" package is used to connect to MongoDB from NodeJS program.
- The "Mongoose schema" defines strcture of the mongodb document.
- The "mongoose model" provides methods for retrieving, inserting, updating, deleting data from database.
- **Steps for working with "mongoose" package:**
  - **Import the package in "package.json" file**

    "dependencies":

    {

```
    "mongoose": "latest"
  }
```

- **Acquire the module:**

  ```
  var mongoose = require("mongoose");
  ```

- **Create mongoose schema:**

  ```
  var schema = new mongoose.Schema( { property: datatype, property: datatype } );
  ```

- **Data types:**

  String, Number, ObjectId, Date, Boolean, Array

- **Create Mongoose Model:**

  ```
  var model = mongoose.model("collection name", Schema);
  ```

- **Connect to database:**

  ```
  mongoose.connect("mongodb://localhost/databasename");
  ```

- **Retrieve all documents:**

  ```
  model.find("callback function);
  ```

- **Insert document:**

  ```
  var newdocument = new model( { property: value, property: value, ...});
  newdocument.save(callbackfunction);
  ```

- **Update document:**

  ```
  existingdocument.save(callbackfunction);
  ```

- **Delete document:**

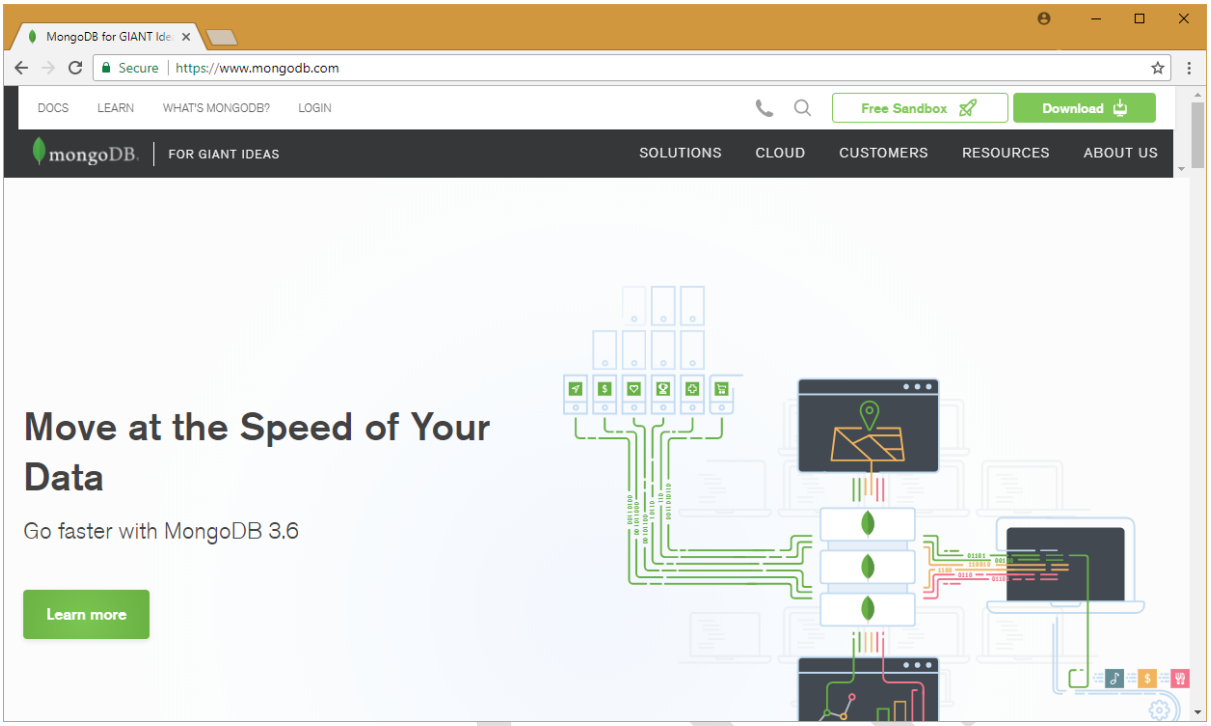
  ```
  model.remove( { condition }, callbackfunction);
  ```

- **Close the connection:**

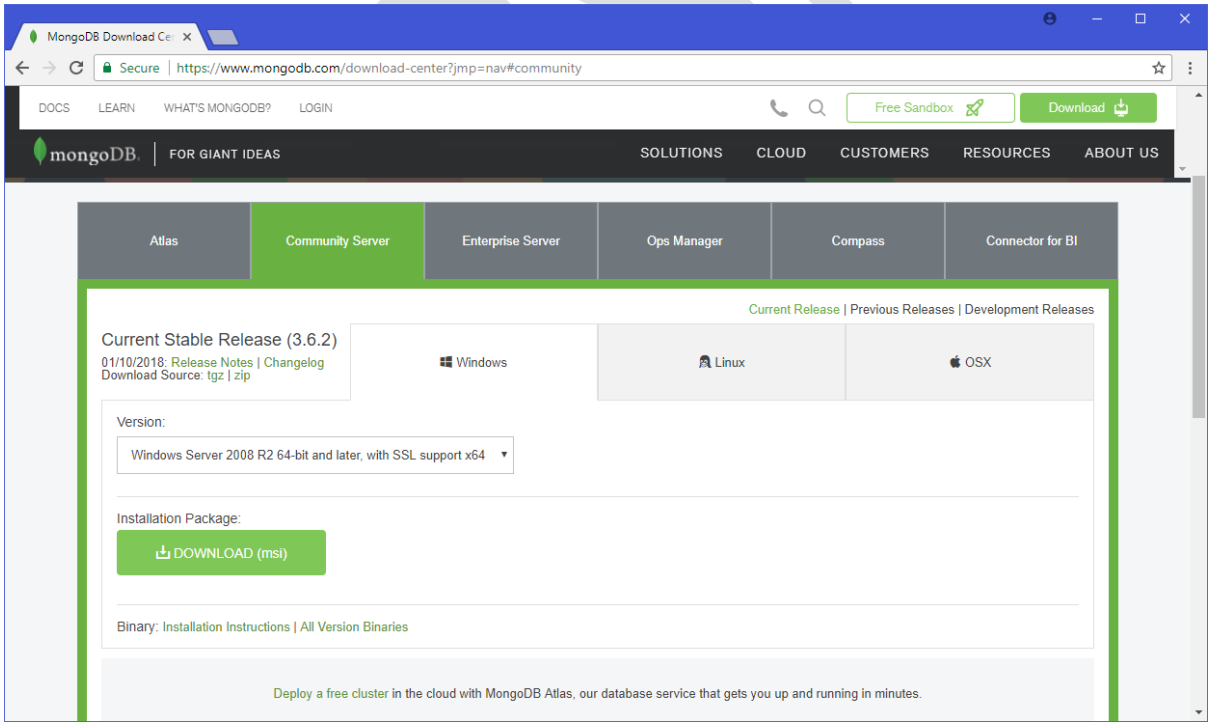
  ```
  mongoose.connection.close();
  ```

Installing MongoDB

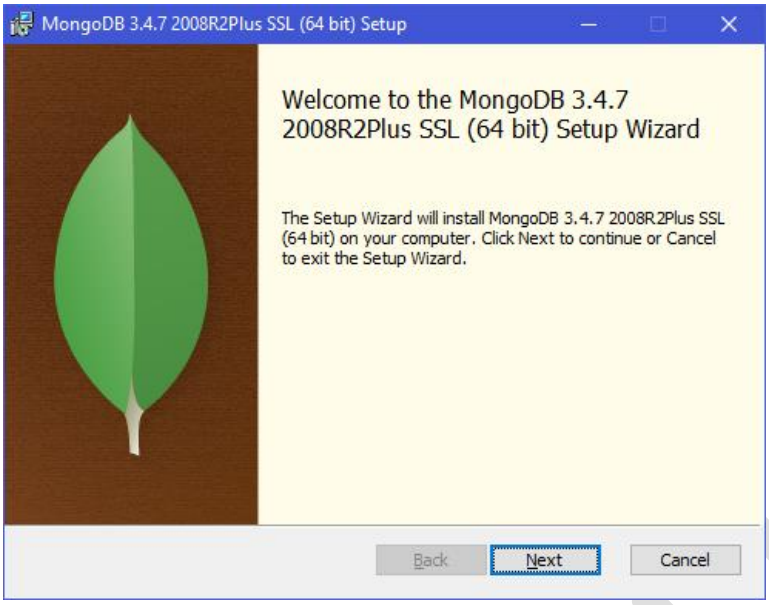
- Go to "https://www.mongodb.com".



- Click on "Download" – "Community Server" – "Windows Server 2008 R2 65-bit and later, with SSL support x64" – "Download (msi)".



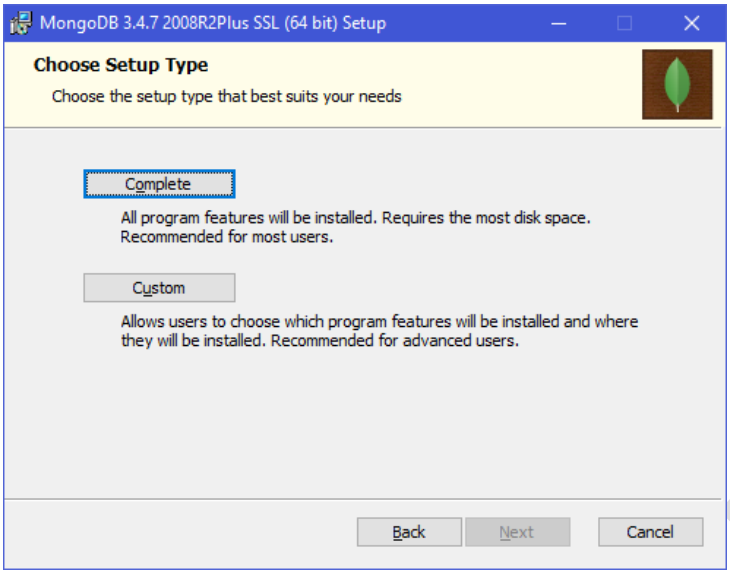
- You will download a file called "mongodb-win32-x86_64-2008plus-ssl-3.4.7-signed.msi".
- Double click on " mongodb-win32-x86_64-2008plus-ssl-3.4.7-signed.msi".



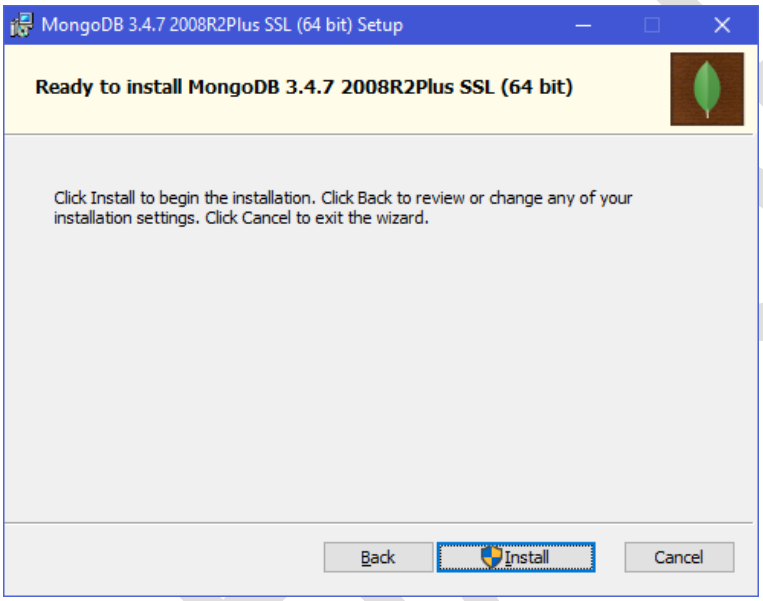
Click on "Next".

Check the checkbox "I accept the terms in the License Agreement".

Click on "Next".

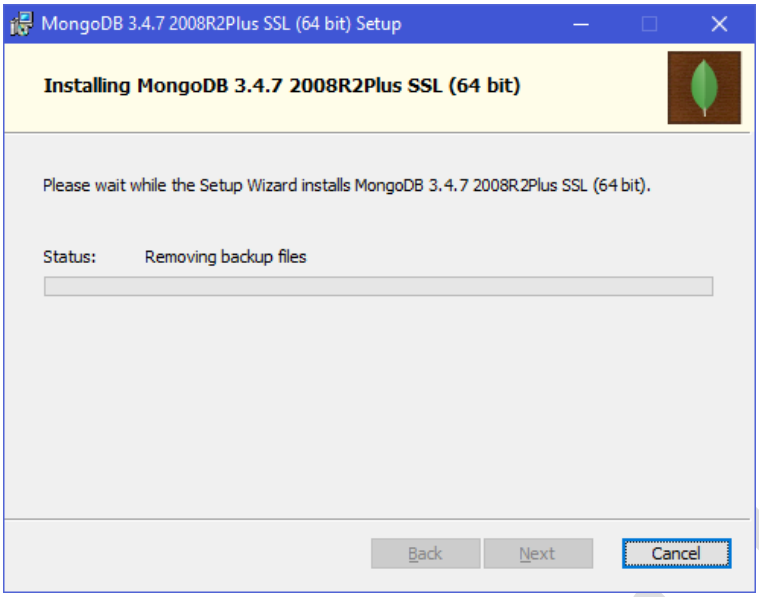


Click on "Complete".

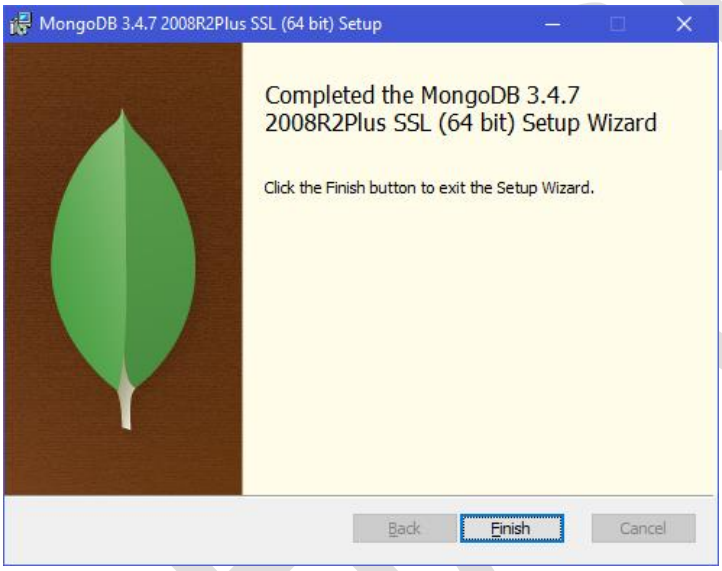


Click on "Install".

Click on "Yes".



Installation is in progress.

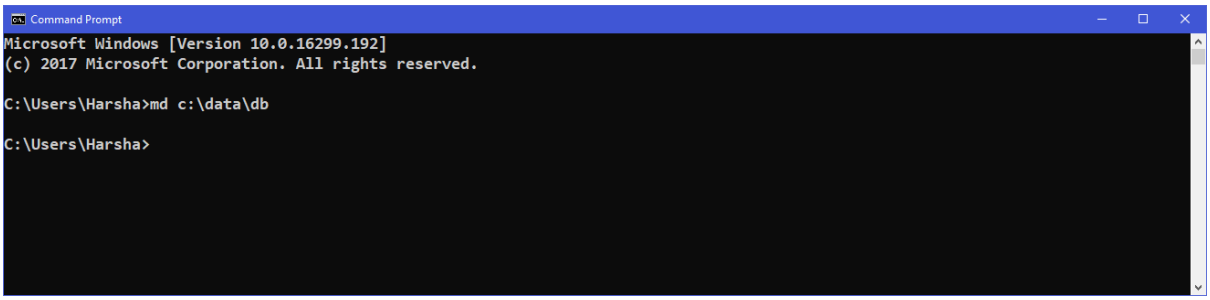


Click on "Finish".

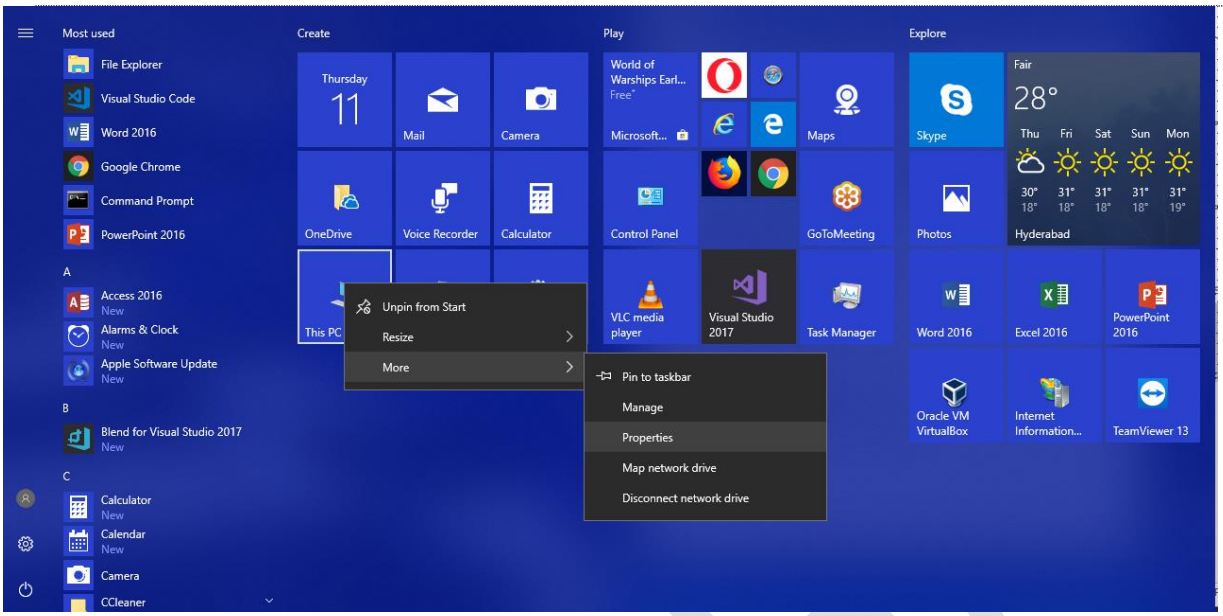
Creating data directory:

Open Command Prompt

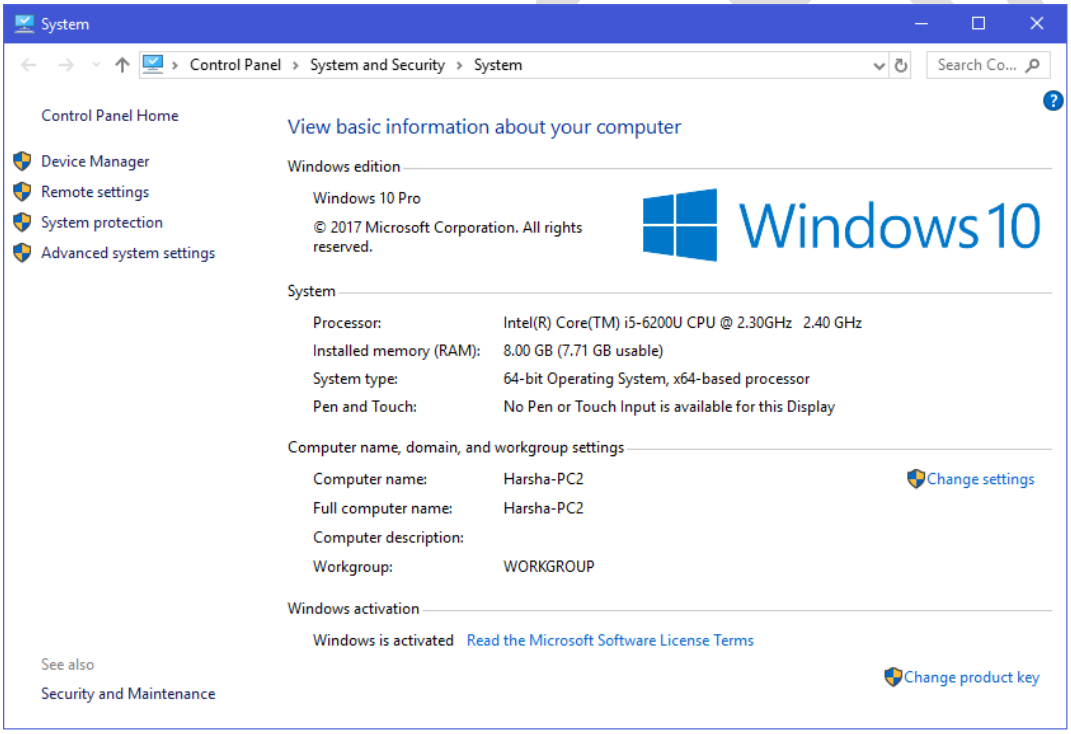
md c:\data\db



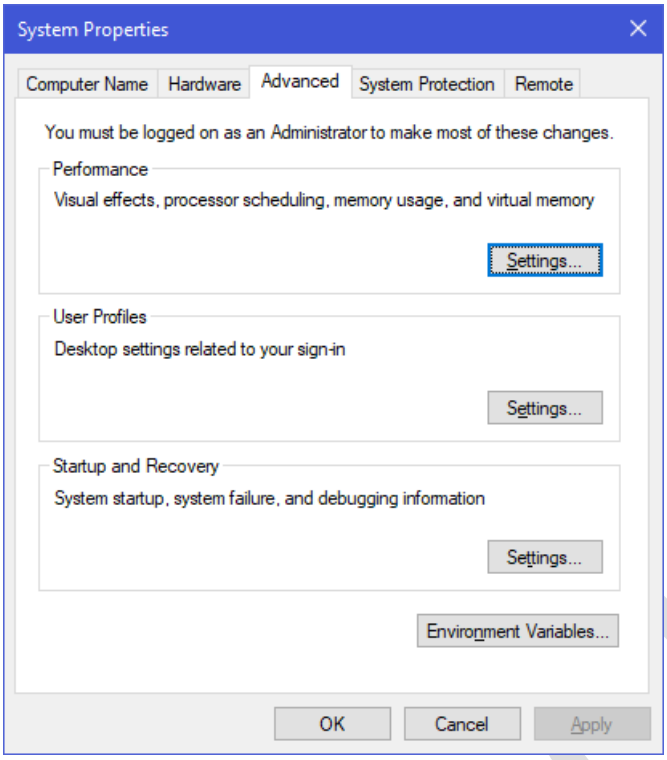
Adding "Path" to Environment Variables



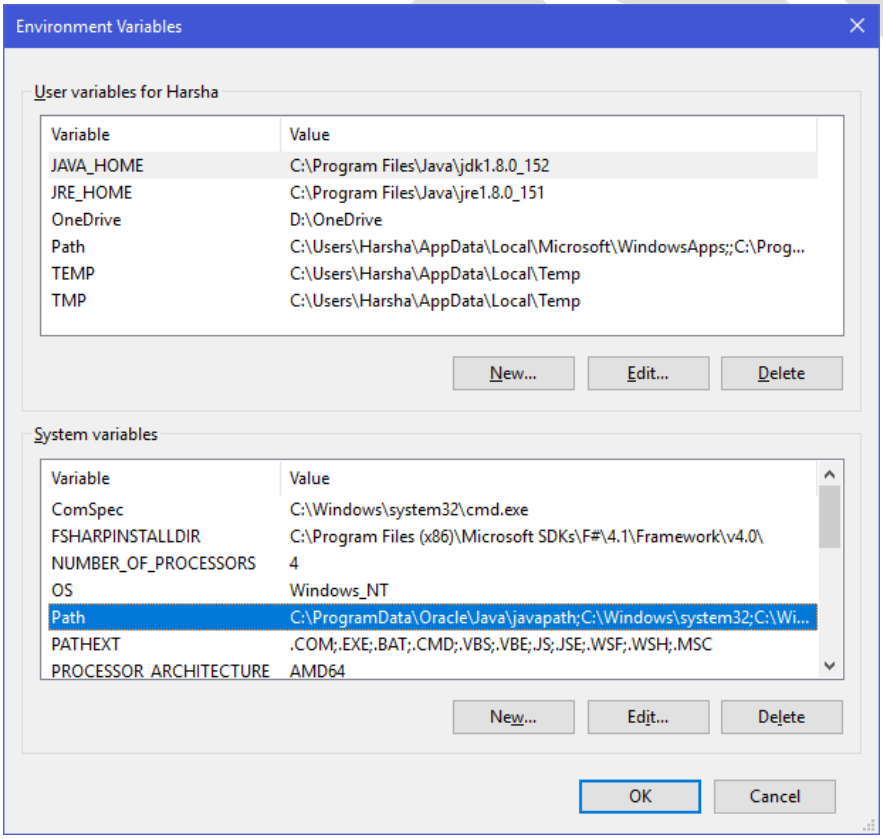
Go to "Start" – Right click on "This PC" – "More" – "Properties".



Click on "Advanced system settings".

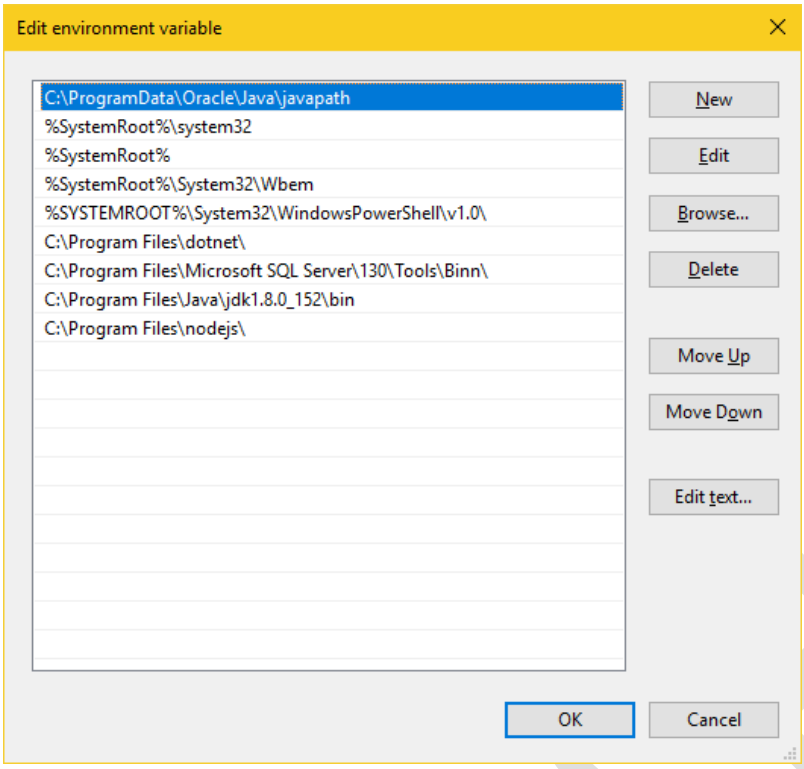


Click on "Environment Variables".

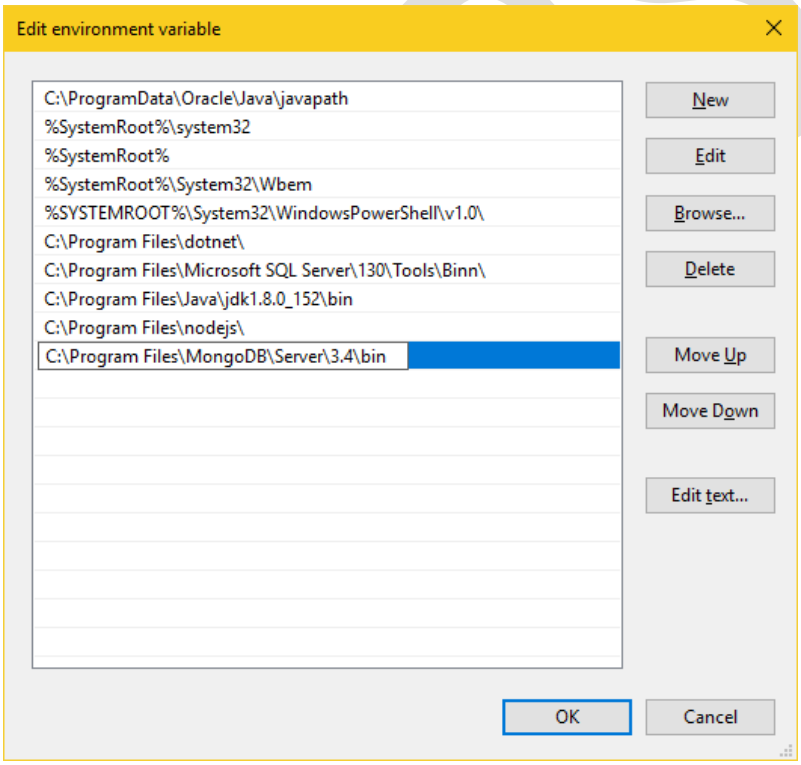


Click on "Path" in "System variables".

Click on "Edit".



Click on "New" and enter the following path: **C:\Program Files\MongoDB\Server\3.4\bin**



Click on OK.

Click on OK again.

Click on OK once again.

Close the System properties.

Find - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "mongoose": "latest"
    }
}
```

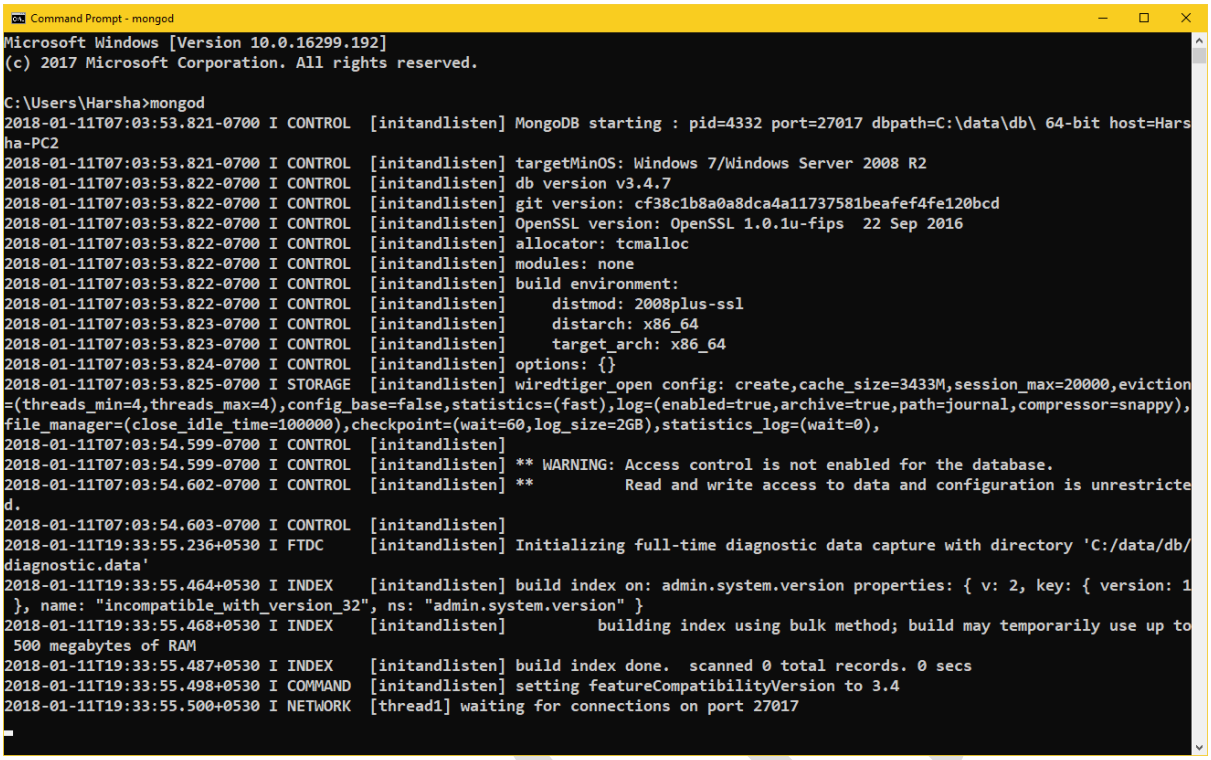
c:\nodejs\find.js

```
var mongoose = require("mongoose");
var EmployeesSchema  = new mongoose.Schema({ empid: Number, empname: String, salary: Number });
var Employee = mongoose.model("employees", EmployeesSchema);
mongoose.connect("mongodb://localhost/company");
//mongoose.connect("mongodb://localhost/company", { user: "username", pass: "password" });

Employee.find(fun1);
function fun1(error, data)
{
    if (error)
    {
        console.log(error);
    }
    else
    {
        console.log(data);
        mongoose.connection.close();
    }
}
```

Execution

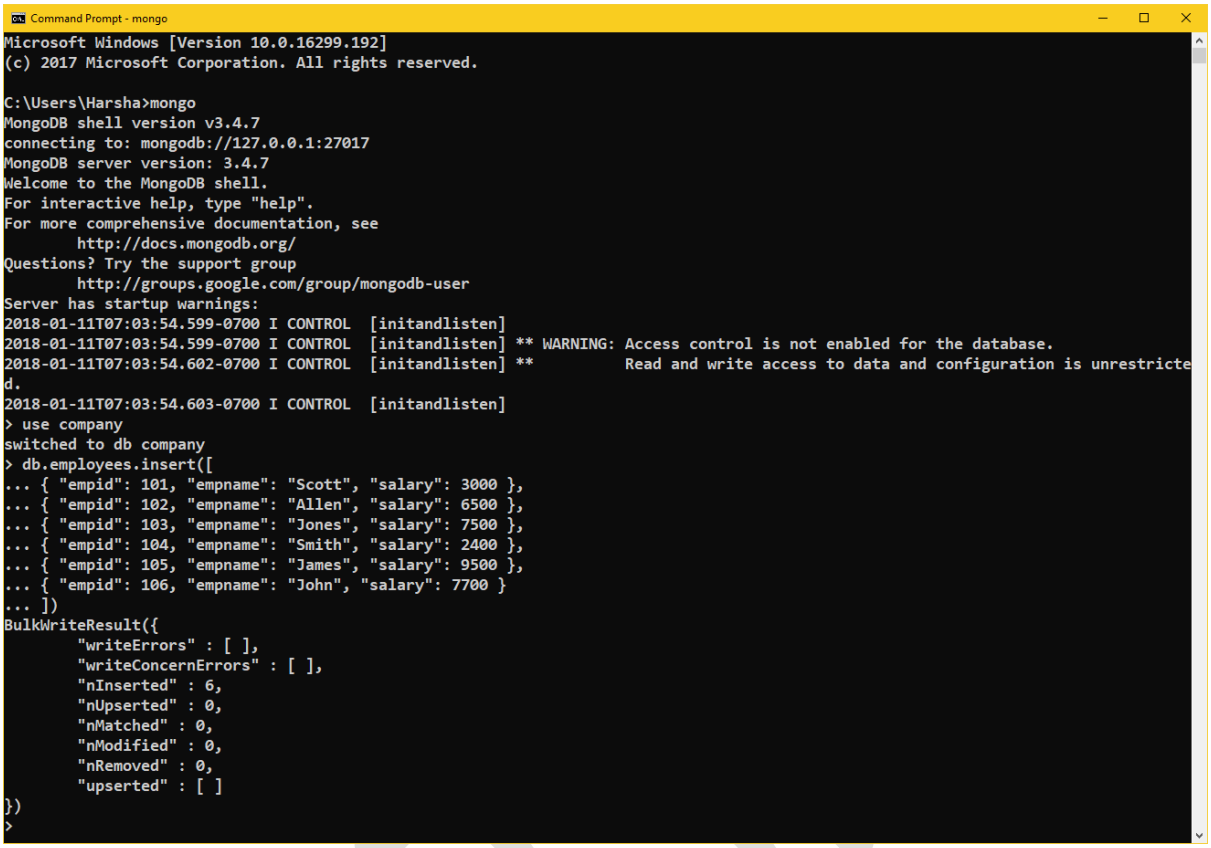
- Open Command Prompt 1

  mongod



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```



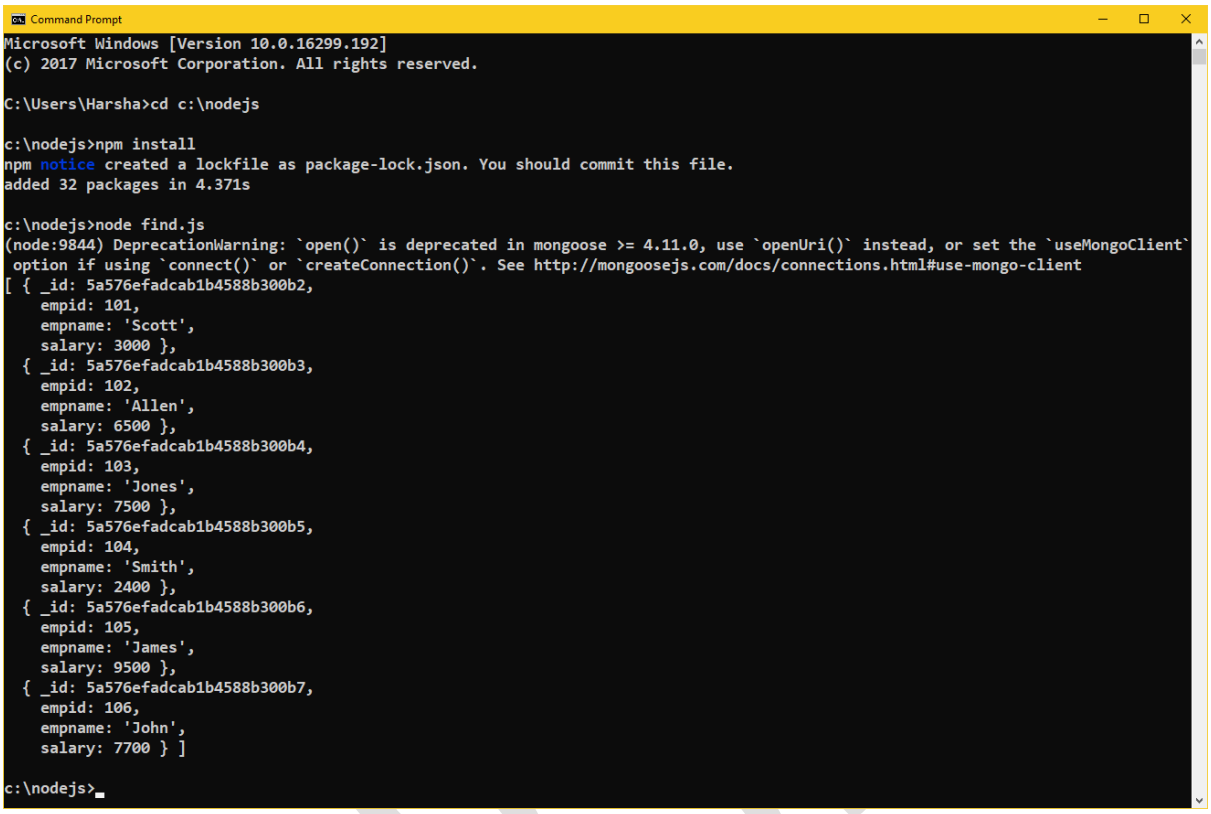
- Open Command Prompt 3

  cd c:\nodejs

  npm install

  node find.js

Output:



Insert - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "mongoose": "latest"
  }
}
```

c:\nodejs\insert.js

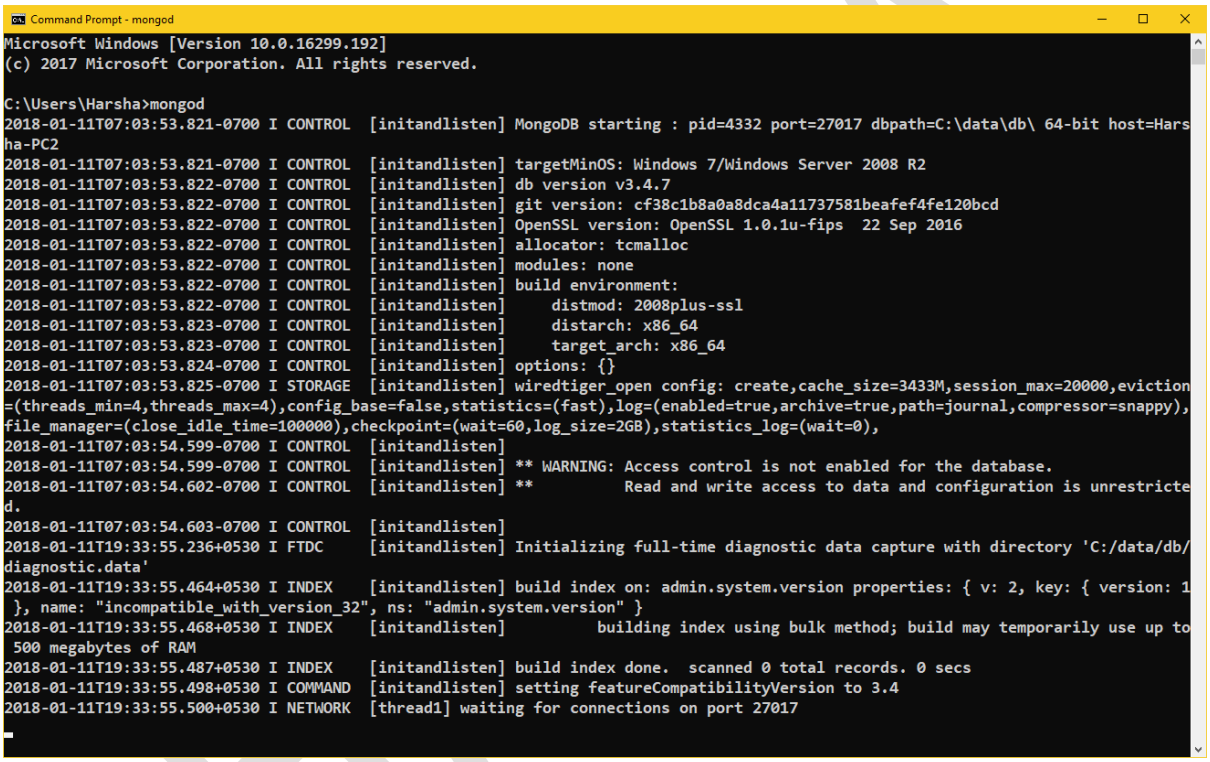
```
var mongoose = require("mongoose");
var EmployeesSchema = new mongoose.Schema({ empid: Number, empname: String, salary: Number }, { versionKey: false });
var Employee = mongoose.model("employees", EmployeesSchema);
mongoose.connect("mongodb://localhost/company");

var newemp = new Employee({ empid: 201, empname: "abc", salary: 10000 });
newemp.save(fun1);
function fun1(error)
{
if (error)
```

```
{
    console.log(error);
}
else
{
    console.log("Successfully inserted");
    mongoose.connection.close();
}
}
```

Execution

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

  mongo

  use company

  db.employees.insert([

  { "empid": 101, "empname": "Scott", "salary": 3000 },

  { "empid": 102, "empname": "Allen", "salary": 6500 },

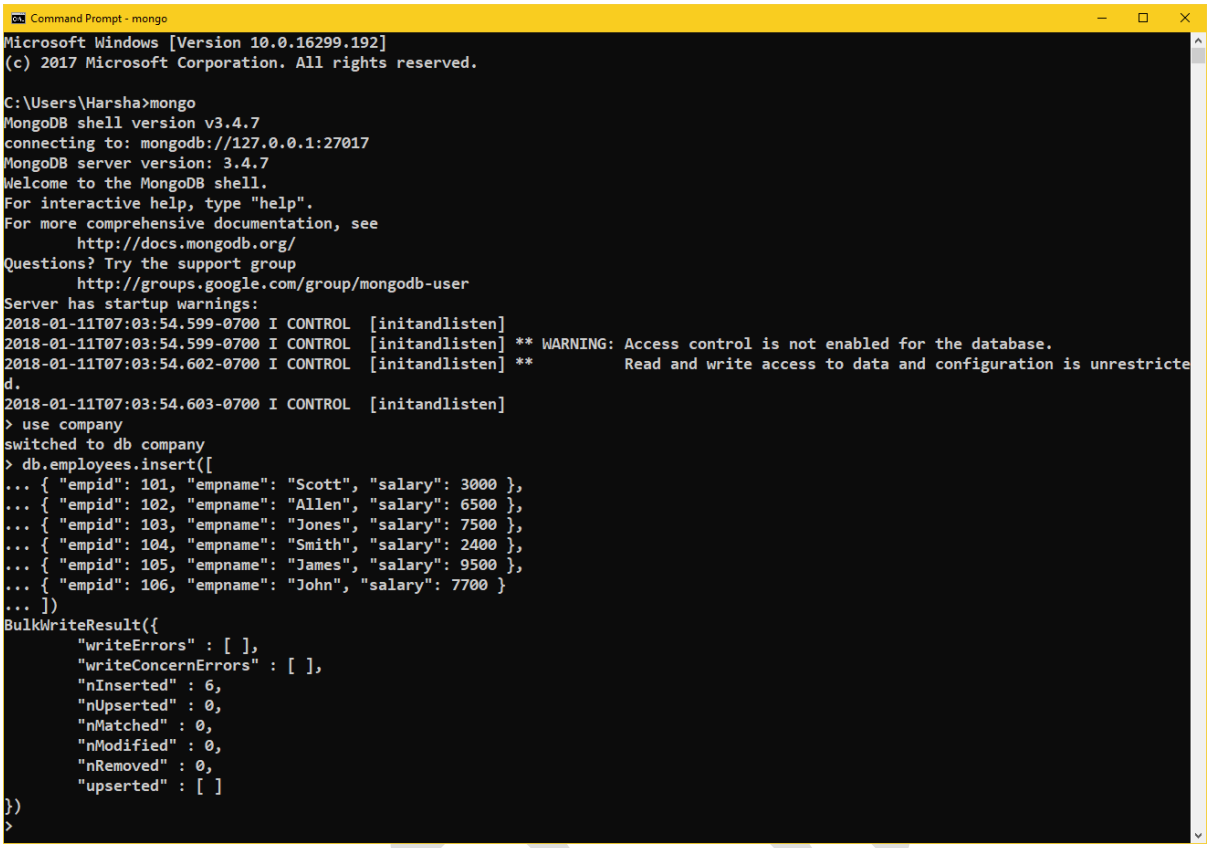
  { "empid": 103, "empname": "Jones", "salary": 7500 },

  { "empid": 104, "empname": "Smith", "salary": 2400 },

  { "empid": 105, "empname": "James", "salary": 9500 },

  { "empid": 106, "empname": "John", "salary": 7700 }

  ])



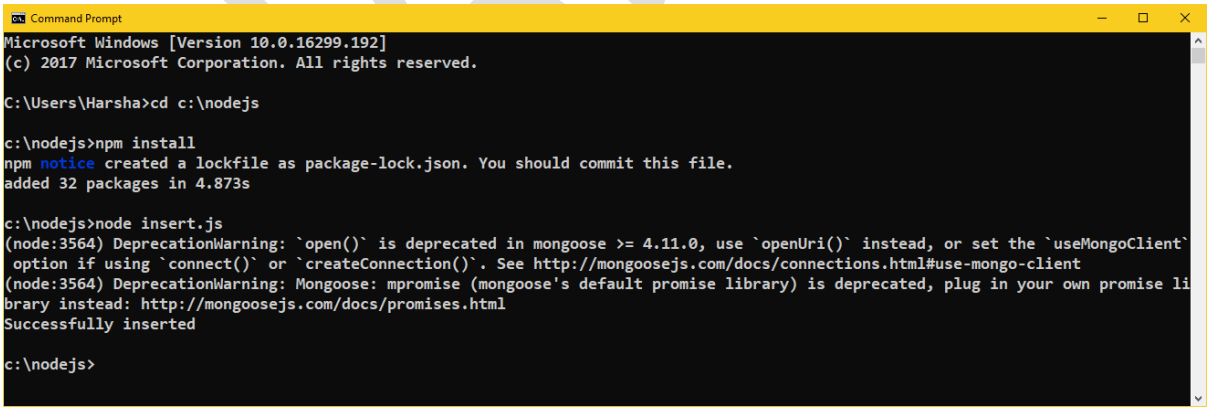
- Open Command Prompt 3

  cd c:\nodejs

  npm install

  node insert.js

Output:



Update - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
```

```
"license": "ISC",
"repository": "none",
"dependencies":
{
    "mongoose": "latest"
}
}
```

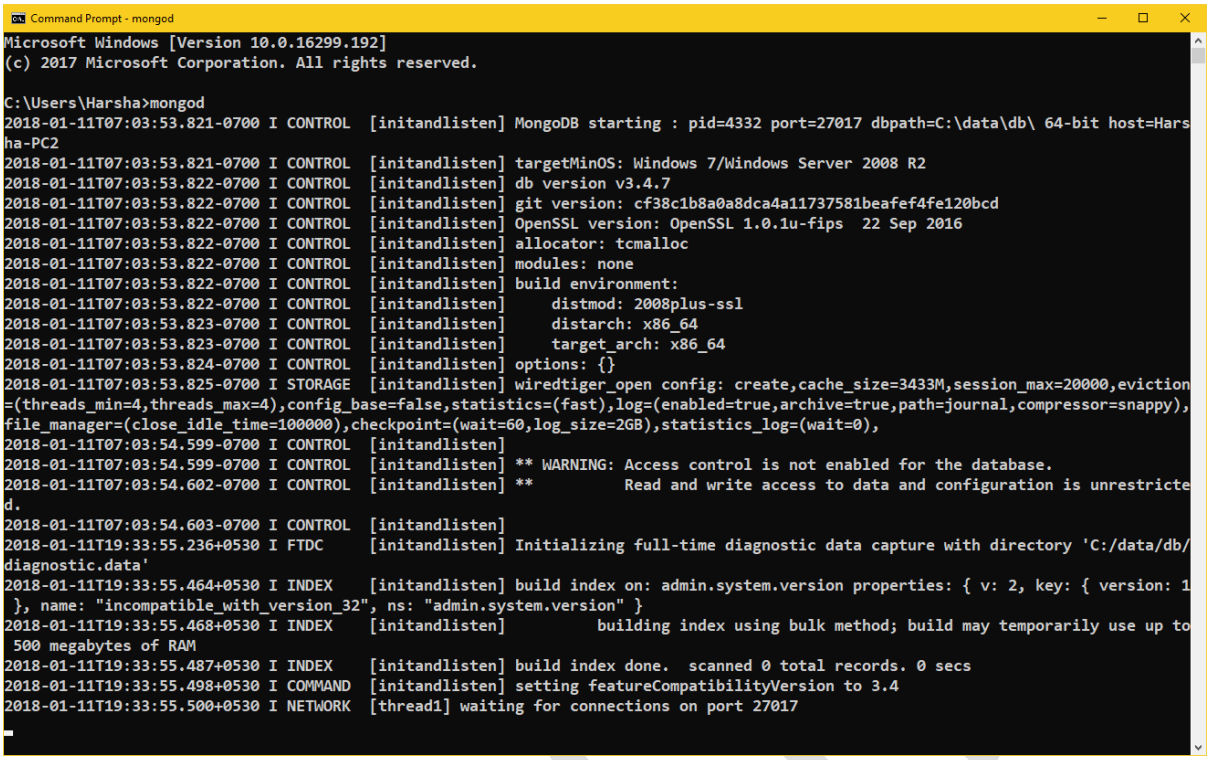
c:\nodejs\update.js

```
var mongoose = require("mongoose");
var EmployeesSchema = new mongoose.Schema({ empid: Number, empname: String, salary: Number }, { versionKey:
false });
var Employee = mongoose.model("employees", EmployeesSchema);
mongoose.connect("mongodb://localhost/company");

Employee.findOne({empid: 201}, fun1);
function fun1(error, existingemp)
{
    if (error)
    {
        console.log("Invalid Emp ID");
    }
    else
    {
        existingemp.empname = "modified";
        existingemp.salary = 16000;
        existingemp.save(fun2);
        function fun2(error)
        {
            if (error)
            {
                console.log(error);
            }
            else
            {
                console.log("Successfully updated");
                mongoose.connection.close();
            }
        }
    }
}
```

Execution

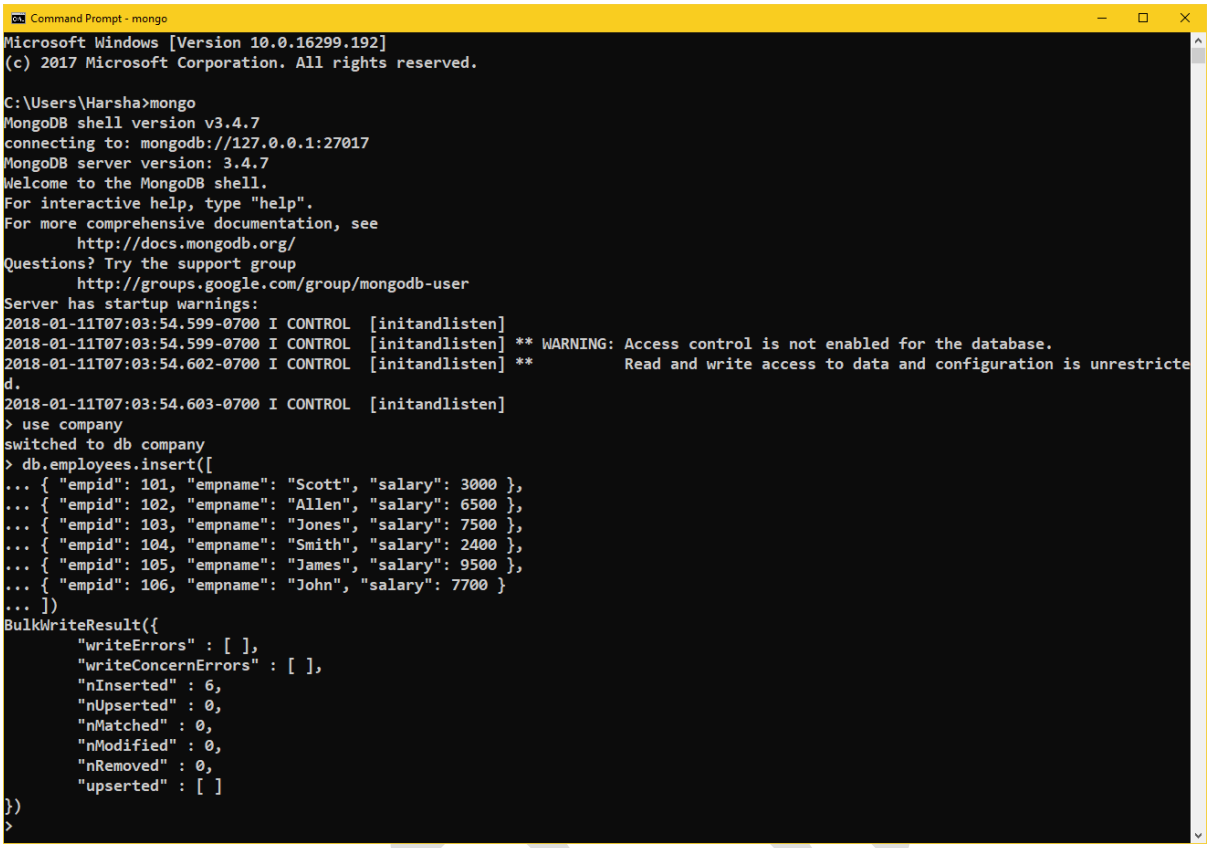
- Open Command Prompt 1

  mongod



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```



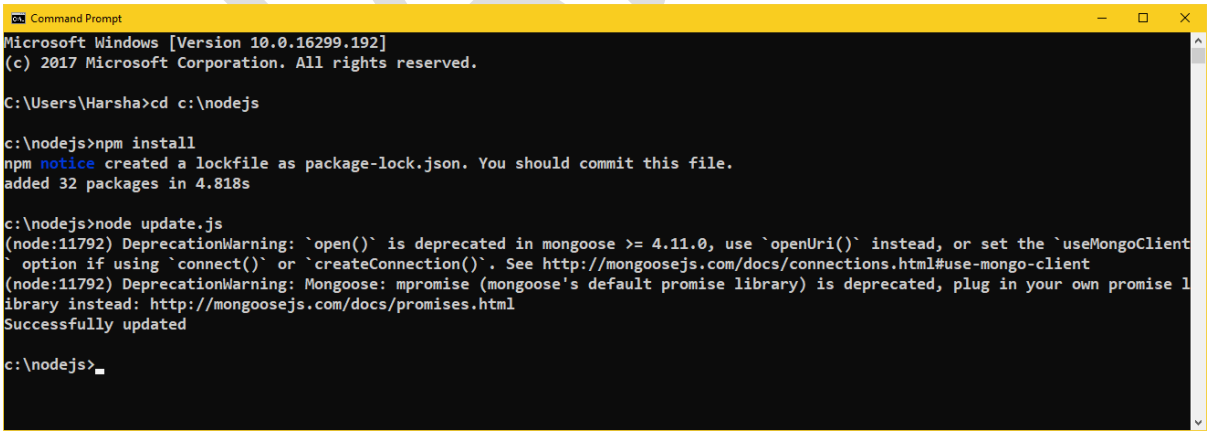
- Open Command Prompt 3

  cd c:\nodejs

  npm install

  node update.js

Output:



Delete - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
```

```
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "mongoose": "latest"
    }
}
```

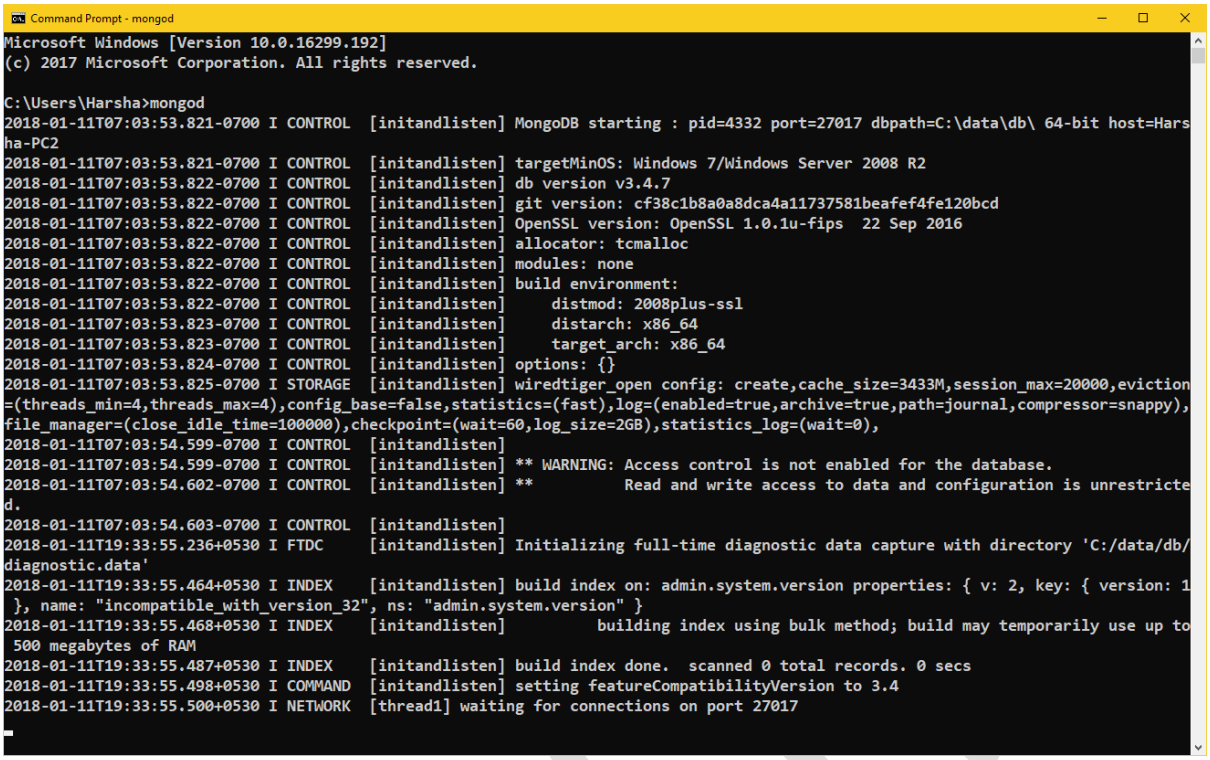
c:\nodejs\delete.js

```
var mongoose = require("mongoose");
var EmployeesSchema = new mongoose.Schema({ empid: Number, empname: String, salary: Number }, { versionKey:
false });
var Employee = mongoose.model("employees", EmployeesSchema);
mongoose.connect("mongodb://localhost/company");

Employee.remove({ empid: 201 }, fun1);
function fun1(error, existingemp)
{
    if (error)
    {
        console.log("Invalid Emp ID");
    }
    else
    {
        console.log("Successfully Deleted");
        mongoose.connection.close();
    }
}
```

Execution

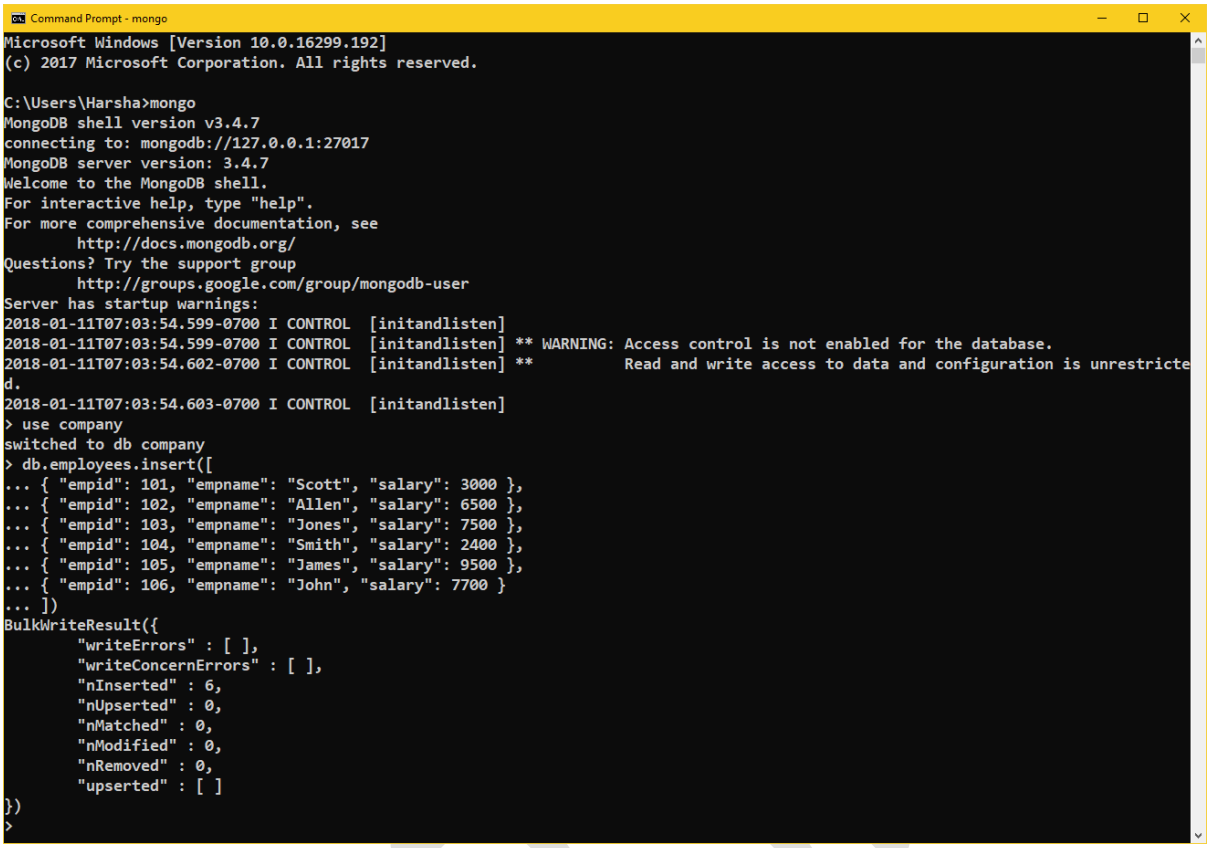
- Open Command Prompt 1

```
mongod
```



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```



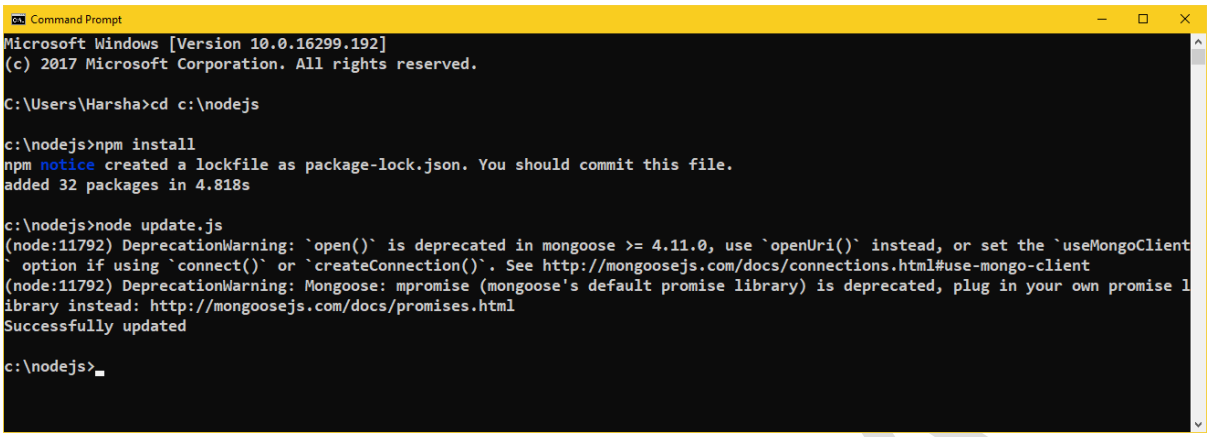
- Open Command Prompt 3

  cd c:\nodejs

  npm install

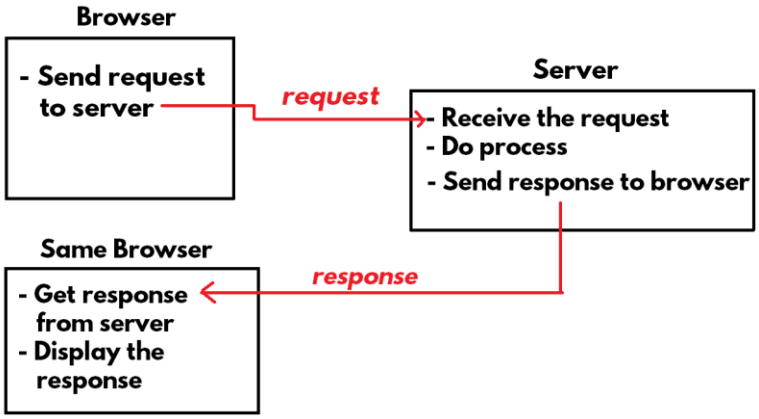
  node update.js

Output:



"http" module

- The "http" module in NodeJS is used to receive the request from browser and send response to the browser.



- **Steps for working with "http" module**
  - **Acquire "http" module:**

    var http = require("http");

  - **Create a http server:**

    var server = http.createServer(engine);

    function engine(request, response)

    {

    //this function executes automatically when the browser sends a request to the server

    }

  - **Start the listener:**

    server.listen(portno, startup);

    function startup()

    {

```
    //This function executes automatically, when the server is started and ready to receie the requests.
}
```

Note: The "portno" can be any number between 1024 and 65535. Ex: 8080

- **Send request from browser**

  Open any browser and enter the following url in the location bar:

  http://localhost:portno

  Ex: http://localhost:8080

Simple Server - Example

c:\nodejs\httpserver.js

```
var http = require("http");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine()
{
    console.log("request received");
}
```

Execution

- Open Command Prompt

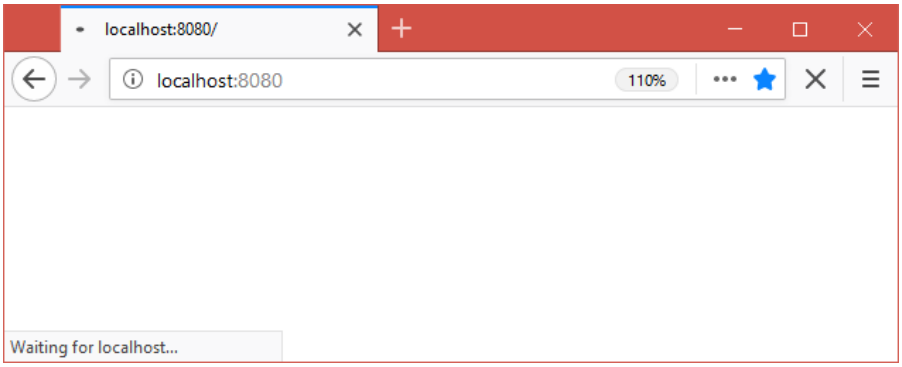
  cd c:\nodejs

  node httpserver.js

- Open any browser

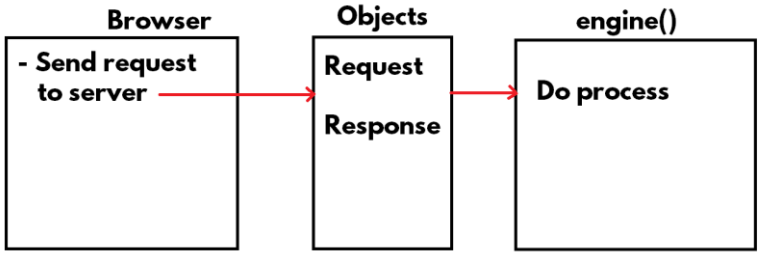
  http://localhost:8080

Output:



Intro to "Request" and "Response" objects

- When the browser sends a request, the server automatically creates two objects:
    - **Request:** Represents the data that is submitted by the browser to server.
    - **Response:** Represents the data that is to be sent to the browser.



Request object

- The "request" object represents the data that is submitted from the browser to the server.
- **Members of "request" object:**
    - **request.url**
        - Represents the url of the current request.
    - **request.method**
        - Represents the type of current request: GET | POST
    - **request.headers**
        - Represents all the request headers that are submitted from browser to server.

Get (vs) Post

- HTTP protocol represents two types of requests:
    1. GET
    2. POST
- **GET:**
    - The "get" type of request is used to retrieve data from server. Ex: google search, Trains search etc.
    - The parameters will be appended to the url as "query string".
      http://localhost:portno?parameter=value&parameter=value...
    - GET is faster than POST
    - GET can pass only ASCII characters; we can't pass Unicode characters through GET.
    - GET can pass limited no. of characters only. Maximum no. of characters per url: 255 characters.
- **POST:**
    - POST is used to perform insert / update / delete operations.
    - The parameters are not sent in the url; but will be sent in "request body".

- The parameters will not be displayed in the address bar of the browser, as they are sent in "request body" (hidden format).
- POST is slower than GET
- POST can send unlimited data
- POST supports Unicode characters.

Request - Example

c:\nodejs\httpserver.js

```
var http = require("http");
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    console.log("request received at " + new Date().toLocaleTimeString());
    console.log("url: " + request.url);
    console.log("method: " + request.method);
    console.log("headers: ");
    console.log(request.headers);
    response.end();
}
```

Execution

- Open Command Prompt

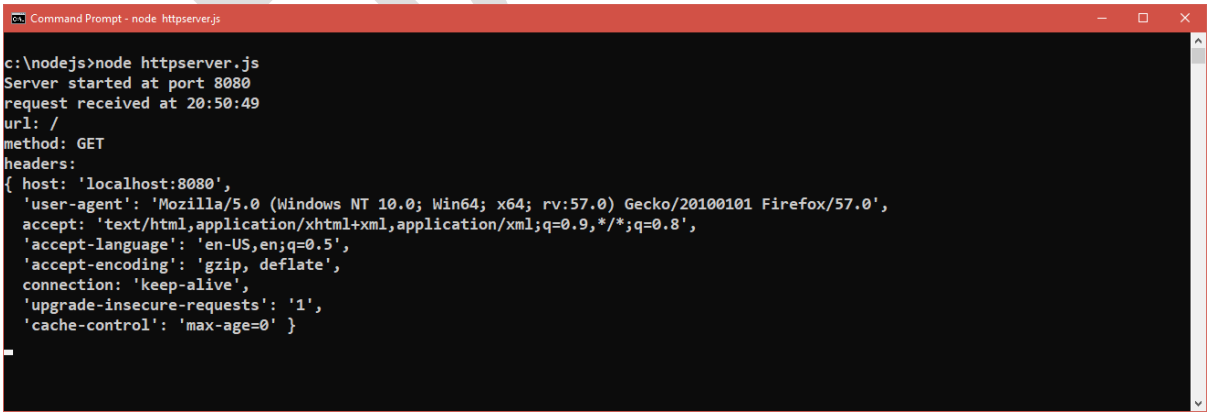
  cd c:\nodejs

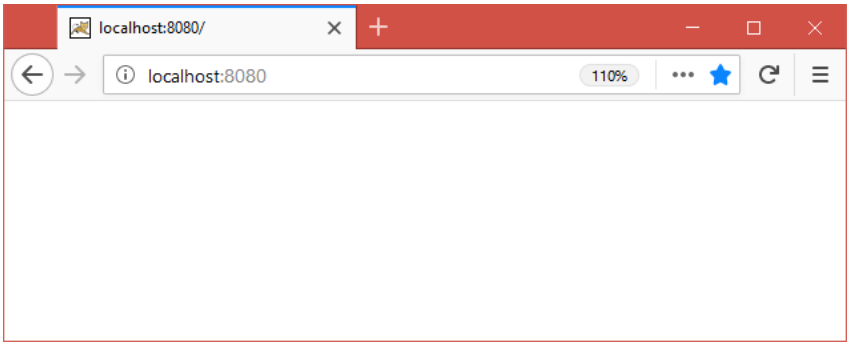
  node httpserver.js

- Open any browser

  http://localhost:8080

Output:





Response object

- The "response" object represents the data that has to be sent from the server to browser.
- **Members of "request" object:**
  - **response.write()**
    - Sends data to browser.
    - **Syntax:** response.write(value);
    - **Example:** response.write("<h1>Hello</h1>");
  - **response.end()**
    - Indicates the browser that the response is finished.
    - **Syntax:** response.end();
    - **Example:** response.end();
  - **response.setHeader()**
    - Adds a header (key with value) to "response headers", that can be sent to the browser.
    - **Syntax:** response.setHeader("key", "value");
    - **Example:** response.setHeader("content-type", "text/html");
    - **List of important content types:**
      - text/plain
      - text/html
      - text/css
      - text/javascript
      - application/json
  - **response.writeHead()**
    - Sets the response status code that can be sent to the browser.
    - **Syntax:** response.writeHead(status code);
    - **Example:** response.writeHead(200);
    - **List of important status codes:**
      - 101 : Information
      - 200 : OK (Success)
      - 302 : Redirection
      - 304 : Not Modified (browser cached)
      - 400 : Bad request
      - 401 : Unauthorized
      - 404 : File not found
      - 500 : Internal Server Error

Response - Example

c:\nodejs\httpserver.js

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

```
var server = http.createServer(engine);
server.listen(8080, startup);

function startup()
{
    console.log("Server started at port 8080");
}

function engine(request, response)
{
    response.setHeader("content-type", "text/html");
    response.writeHead(200);
    response.write("<h1>Hello</h1>");
    response.end();
}
```

Execution

- Open Command Prompt

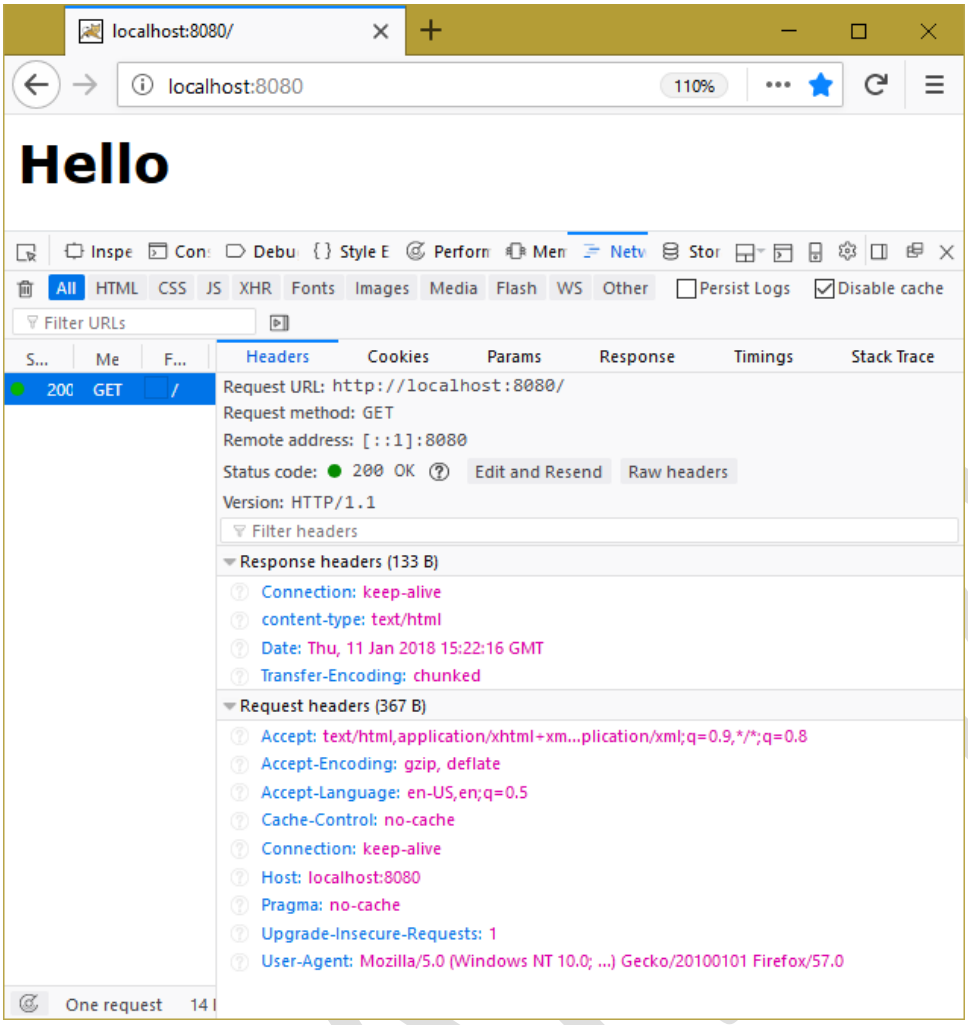
  cd c:\nodejs

  node httpserver.js

- Open any browser

  http://localhost:8080

Output:



Static Pages - Example

c:\nodejs\httpserver.js

```
//http
var http = require("http");
var server = http.createServer(engine);
server.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//fs
var fs = require("fs");

//engine
function engine(request, response)
{
    console.log("request received at " + request.url);

    var filename = "";
    if (request.url == "/" || request.url == "/home")
```

```
            filename = "home.html";
        else if (request.url == "/about")
            filename = "about.html";
        else if (request.url == "/contact")
            filename = "contact.html";
        else
            filename = "404.html";

        console.log(filename);
        fs.readFile(filename, "utf8", fun1);

        function fun1(error, data)
        {
            if (error != null)
            {
                console.log(error);
                response.setHeader("content-type", "text/html");
                response.writeHead(500);
                response.write("<h1 style='color:red'>Unable to load data</h1>");
                response.end();
            }
            else
            {
                //console.log(data);
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
```

c:\nodejs\home.html

```
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        <div id="nav">
            <a href="/home">Home</a>
            <a href="/about">About</a>
            <a href="/contact">Contact</a>
        </div>
        <h1>Home page content here</h1>
    </body>
</html>
```

c:\nodejs\about.html

```
<html>
    <head>
        <title>About</title>
    </head>
    <body>
        <div id="nav">
            <a href="/home">Home</a>
```

```
            <a href="/about">About</a>
            <a href="/contact">Contact</a>
        </div>
        <h1>About page content here</h1>
    </body>
</html>
```

c:\nodejs\contact.html

```
<html>
    <head>
        <title>Contact</title>
    </head>
    <body>
        <div id="nav">
            <a href="/home">Home</a>
            <a href="/about">About</a>
            <a href="/contact">Contact</a>
        </div>
        <h1>Contact page content here</h1>
    </body>
</html>
```

Execution

- Open Command Prompt

  cd c:\nodejs

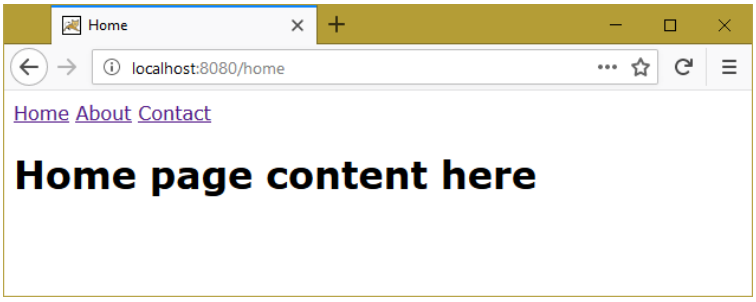
  node httpserver.js

- Open any browser

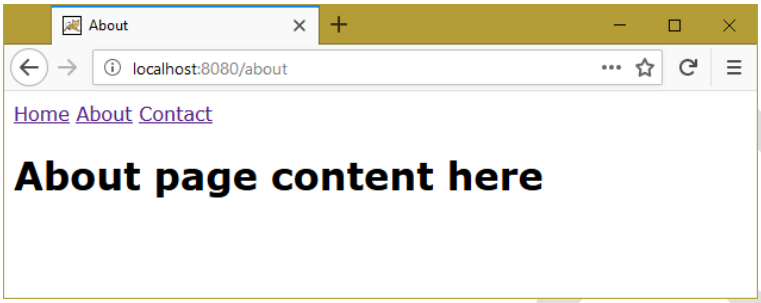
  http://localhost:8080

Output:

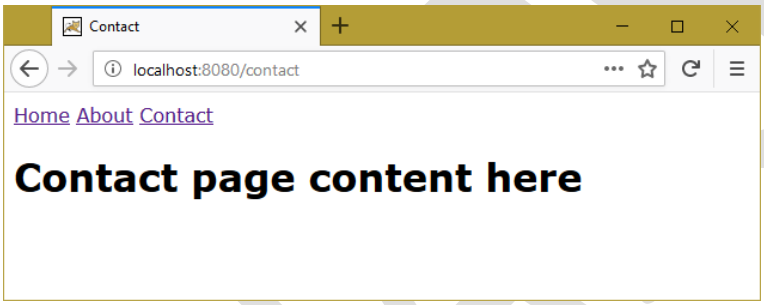
http://localhost:8080/home



http://localhost:8080/about



http://localhost:8080/contact



"querystring" Module

- The "querystring" module is used to parse (convert) the "querystring parameters" into "javascript object" format.

  Browser → Request → Express → querystring → Convert querystring to js object

- **Steps for working with "querystring" module**
  - **Acquire "querystring" module:**

    var querystring = require("querystring");

  - **Parse the "querystring parameters" into "javascript object":**

    var q = querystring.parse(request.url.split("?")[1]);

  - **Get the value of individual parameter:**

    q.parametername

Forms - Get - Example

c:\nodejs\httpserver.js

```
//http
var http = require("http");
var server = http.createServer(engine);
server.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//fs
var fs = require("fs");

//querystring
var querystring = require("querystring");

//engine
function engine(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);

    if (request.url == "/")
    {
        fs.readFile("index.html", "utf8", fun1);

        function fun1(error, data)
        {
            if (error)
            {
                console.log(error);
                response.setHeader("content-type", "text/html");
                response.writeHead(404);
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.setHeader("content-type", "text/html");
                response.writeHead(200);
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url.startsWith("/search"))
    {
        var q = querystring.parse((request.url.split('?')[1]));
        console.log(q);
        response.setHeader("content-type", "text/html");
        response.writeHead(200);
        response.write("<h1>Search results for: " + q.searchbox + "</h1>");
        response.end();
    }
}
```

c:\nodejs\index.html

```
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        <form action="/search" method="get">
            Type your search:
            <input type="text" name="searchbox"><br>
            <input type="submit" value="Search">
        </form>
    </body>
</html>
```

Execution

- Open Command Prompt

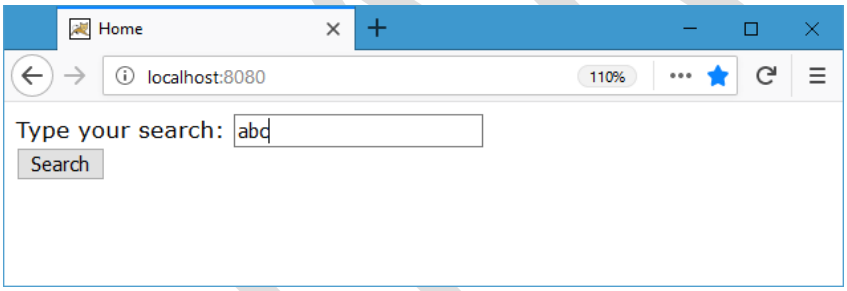
  cd c:\nodejs

  node httpserver.js

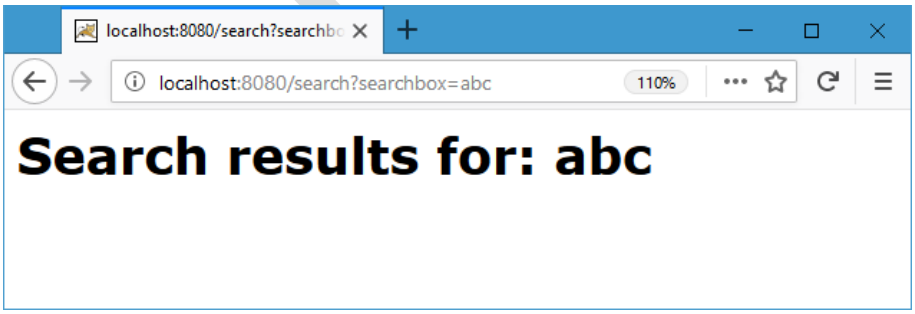
- Open any browser

  http://localhost:8080

Output:



Enter some text and click on "Search".



Forms - Post - Example

c:\nodejs\httpserver.js

```
//http
var http = require("http");
var server = http.createServer(engine);
server.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//fs
var fs = require("fs");

//querystring
var querystring = require("querystring");

//engine
function engine(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);

    if (request.url == "/" && request.method =="GET")
    {
        fs.readFile("login.html", "utf8", fun1);
        function fun1(error, data)
        {
            if (error)
            {
                console.log(error);
                response.writeHead(404, { "content-type": "text/html" });
                response.write("<h1 style='color:red'>Page not found</h1>");
                response.end();
            }
            else
            {
                response.writeHead(200, { "content-type": "text/html" });
                response.write(data);
                response.end();
            }
        }
    }
    else if (request.url.startsWith("/login") && request.method == "POST")
    {
        request.on("data", fun2);
        function fun2(data)
        {
            console.log(data.toString());
            var q = querystring.parse(data.toString());
            console.log(q);
            response.writeHead(200, { "content-type": "text/html" });

            if (q.uname == "admin" && q.pwd == "server")
            {
                response.write("<h1 style='color:green'>Successful login</h1>");
```

```
            }
            else
            {
                response.write("<h1 style='color:red'>Invalid login</h1>");
            }
            response.end();
        }
    }
}
```

c:\nodejs\login.html

```
<html>
    <head>
        <title>Login</title>
    </head>
    <body>
        <form action="/login" method="post">
            Username: <input type="text" name="uname"><br>
            Password: <input type="password" name="pwd"><br>
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

Execution

- Open Command Prompt

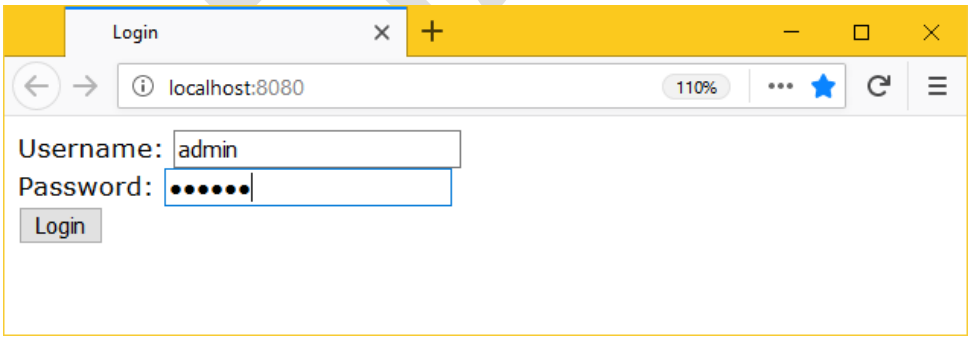
  cd c:\nodejs

  node httpserver.js

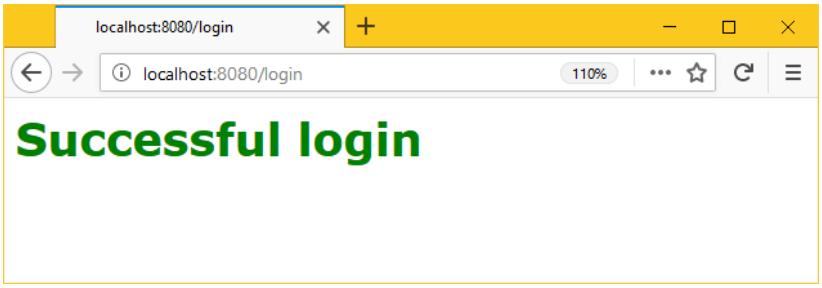
- Open any browser

  http://localhost:8080

Output:



Enter "admin" and "server"; click on "Login".



Drawbacks of Http Module

- Single function for all url's; Expected to have separate functions for each url.
- It is not possible to pass js object to html file.
- MVC architecture not supported.

Solution: "Express" Framework

Express Package

- The "express" package makes it easy to handle the request and response.
- **Advantages of "express" package:**
  - Easy to handle request and response. It provides high-level API, which wraps "http" module.
  - We can pass js object to html file.
  - We can implement "MVC architecture".

Steps for working with "express" package

- **Import "express" package in "package.json" file:**

```
"dependencies":
{
 "express": "latest"
}
```

- **Acquire "express" module:**

```
var express = require("express");
```

- **Enable "express" in the application:**

```
var app = express();
```

- **Start the listener:**

```
app.listen(port no, startup);
function startup()
{
 console.log("Server started");
}
```

  Note: The startup() function executes when the server is started and ready to accept the requests.

- **Receive any request:**

```
app.use(callbackfunction);
function callbackfunction(request, response)
{
}
```

Note: The callbackfunction executes automatically, when the browser sends any request to server. It receives two arguments; the first one is "request", which represents the data that is sent from browser to server. The second one is "response", which represents the data that is to be sent from server to browser. The callback function name, request and response arguments name can be anything.

- **Receive "get" request:**

```
app.get("url", callbackfunction);
function callbackfunction(request, response)
{
}
```

Note: The callbackfunction executes automatically, when the browser sends "get" request to the specific url. It receives two arguments; the first one is "request", which represents the data that is sent from browser to server. The second one is "response", which represents the data that is to be sent from server to browser. The callback function name, request and response arguments name can be anything.

- **Receive "post" request:**

```
app.post("url", callbackfunction);
function callbackfunction(request, response)
{
}
```

Note: The callbackfunction executes automatically, when the browser sends "post" request to the specific url. It receives two arguments; the first one is "request", which represents the data that is sent from browser to server. The second one is "response", which represents the data that is to be sent from server to browser. The callback function name, request and response arguments name can be anything.

Request object in "express" package

- The "request" object represents the data that is submitted from the browser to the server. The "request" object of "express" package is advanced than http module's request object; The "request" object of "express" package contains some additional members, which are not available in http module.
- **Members of "request" object of "express" package:**
  1. **request.url**
     - Represents the url of the current request.
  2. **request.method**
     - Represents the type of current request: GET | POST
  3. **request.headers**
     - Represents all the request headers that are submitted from browser to server.
  4. **request.query**
     - Represents all the "querystring parameters" that are submitted by the browser. It represents the parameters as a Javascript object.

Response object in "express" package

- The "response" object represents the data that has to be sent from the server to browser. The "response" object of "express" package is advanced than http module's response object. The "response" object of "express" package contains some additional members, which are not available in http module.
- **Members of "response" object of "express" package:**
  - **response.write()**
    - Sends data to browser.
    - **Syntax:** response.write(value);
    - **Example:** response.write("<h1>Hello</h1>");
  - **response.end()**
    - Indicates the browser that the response is finished.
    - **Syntax:** response.end();
    - **Example:** response.end();

- **response.header()**
  - Adds a header (key with value) to "response headers", that can be sent to the browser.
  - **Syntax:** response.header("key", "value");
  - **Example:** response.header("content-type", "text/html");
  - **List of important content types:**
    - text/plain
    - text/html
    - text/css
    - text/javascript
    - application/json

- **response.status()**
  - Sets the response status code that can be sent to the browser.
  - **Syntax:** response.status(status code);
  - **Example:** response.status(200);
  - **List of important status codes:**
    - 101 : Information
    - 200 : OK (Success)
    - 302 : Redirection
    - 304 : Not Modified (browser cached)
    - 400 : Bad request
    - 401 : Unauthorized
    - 404 : File not found
    - 500 : Internal Server Error

- **response.send()**
  - Sets the data to browser and also ends the response.
  - **Syntax:** response.end(status code);
  - **Example:** response.end(200);

- **response.sendFile()**
  - Reads the content of specified file, sends the same as response to browser and also ends the response.
  - **Syntax:** response.sendFile("filepath");
  - **Example:** response.sendFile(__dirname + "index.html");

Simple Server - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "express": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
```

```
{
    console.log("Server started at port 8080");
}

app.use(fun1);
function fun1(request, response)
{
    console.log("request received");
    response.write("<h1>Hello World</h1>");
    response.end();
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

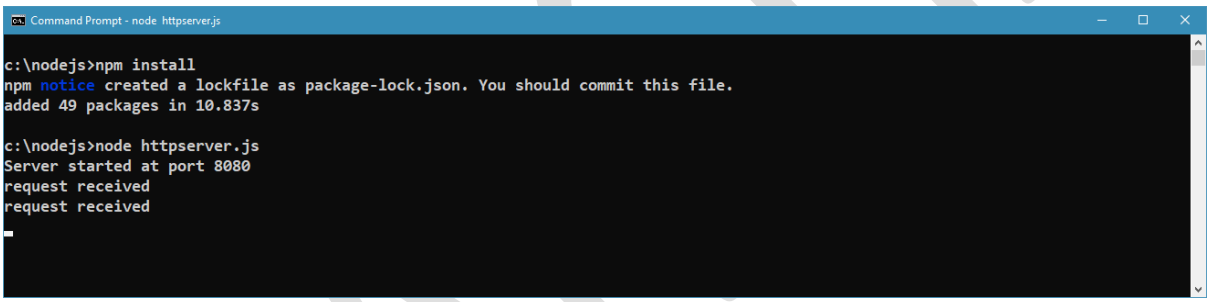
  npm install

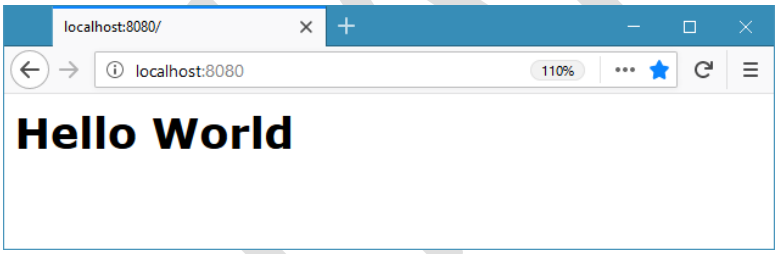
  node httpserver.js

- Open any browser

  http://localhost:8080

Output:





Request Object - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "express": "latest"
    }
}
```

c:\nodejs\httpserver.js

```
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

app.use(fun1);
function fun1(request, response)
{
    console.log("request received at " + new Date().toLocaleTimeString());
    console.log("url: " + request.url);
    console.log("method: " + request.method);
    console.log("headers: ");
    console.log(request.headers);
    console.log("querystring: ");
    console.log(request.query);
    response.end();
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

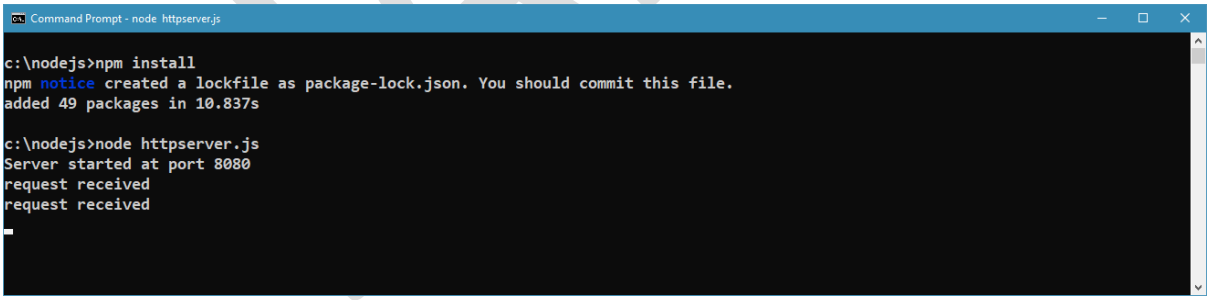
  npm install

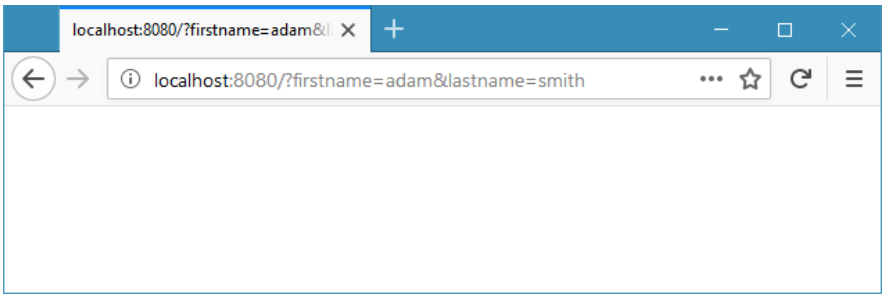
  node httpserver.js

- Open any browser

  http://localhost:8080/?firstname=adam&lastname=smith

Output:





Response Object - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "express": "latest"
    }
}
```

c:\nodejs\httpserver.js

```
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

app.use(fun1);
function fun1(request, response)
{
    response.header("content-type", "text/html");
    response.status(500);
    response.send("<h1>Hello</h1>");
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

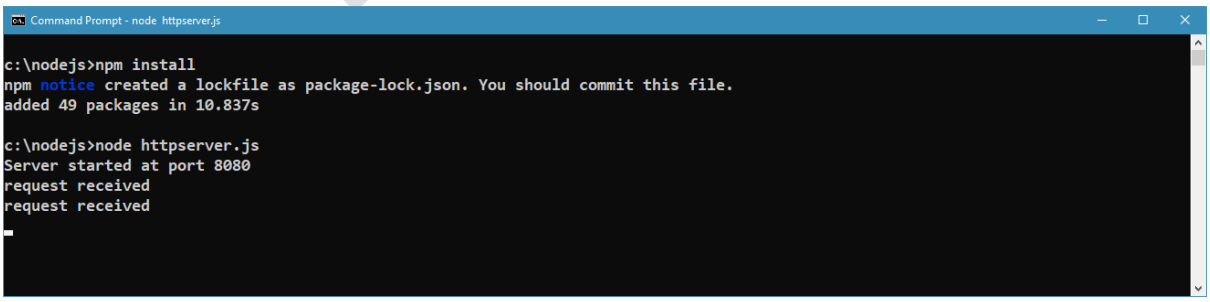
  npm install

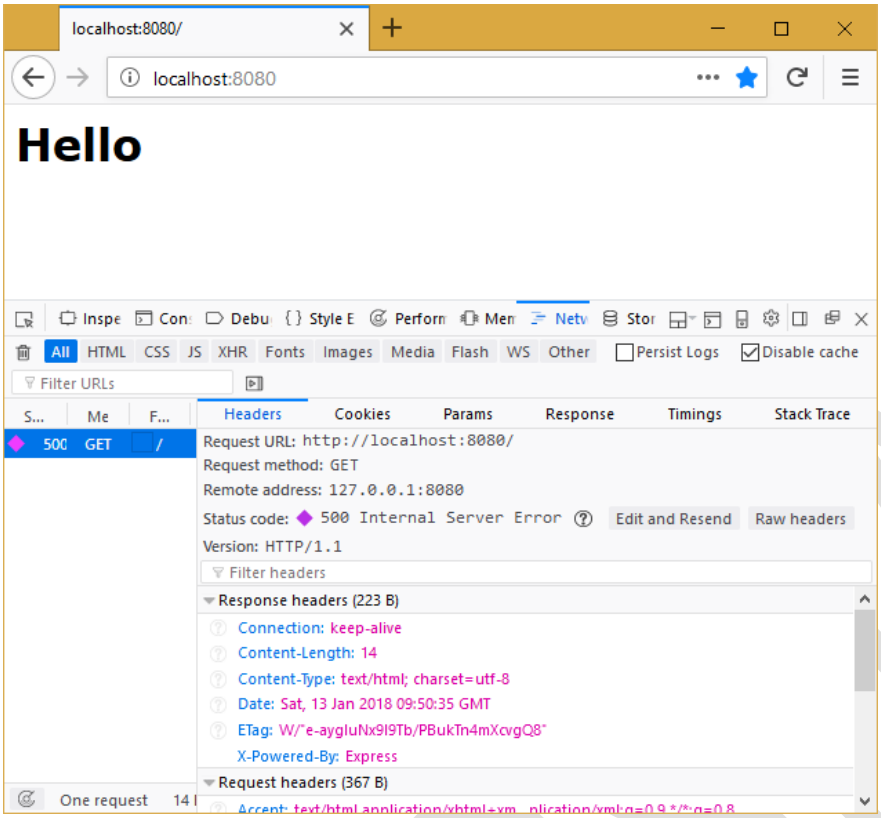
  node httpserver.js

- Open any browser

  http://localhost:8080

Output:





Static Pages - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "express": "latest"
    }
}
```

c:\nodejs\home.html

```
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        <div id="nav">
            <a href="/home">Home</a>
            <a href="/about">About</a>
            <a href="/contact">Contact</a>
        </div>
        <h1>Home page content here</h1>
    </body>
```

```
</html>
```

c:\nodejs\about.html

```
<html>
    <head>
        <title>About</title>
    </head>
    <body>
        <div id="nav">
            <a href="/home">Home</a>
            <a href="/about">About</a>
            <a href="/contact">Contact</a>
        </div>
        <h1>About page content here</h1>
    </body>
</html>
```

c:\nodejs\contact.html

```
<html>
    <head>
        <title>Contact</title>
    </head>
    <body>
        <div id="nav">
            <a href="/home">Home</a>
            <a href="/about">About</a>
            <a href="/contact">Contact</a>
        </div>
        <h1>Contact page content here</h1>
    </body>
</html>
```

c:\nodejs\404.html

```
<html>
    <head>
        <title>Page not found</title>
    </head>
    <body>
        <h1 style='color:red'>Sorry, page not found</h1>
        <a href="/home">Go to home</a>
    </body>
</html>
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//home
app.get("/", fun1);
```

```
app.get("/home", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url);
    response.status(500);
    response.sendFile(__dirname + "/home.html");
}

//about
app.get("/about", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url);
    response.sendFile(__dirname + "/about.html");
}

//contact
app.get("/contact", fun3);
function fun3(request, response)
{
    console.log("request received at " + request.url);
    response.sendFile(__dirname + "/contact.html");
}

//error
app.use(fun4);
function fun4(request, response)
{
    response.sendFile(__dirname + "/404.html");
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

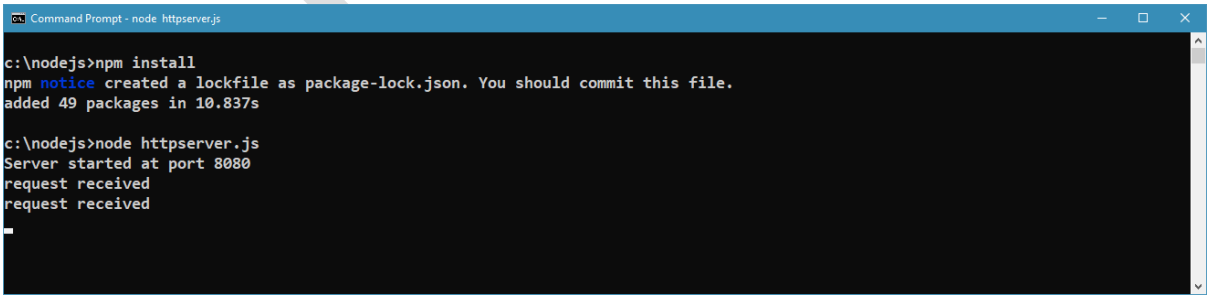
  npm install

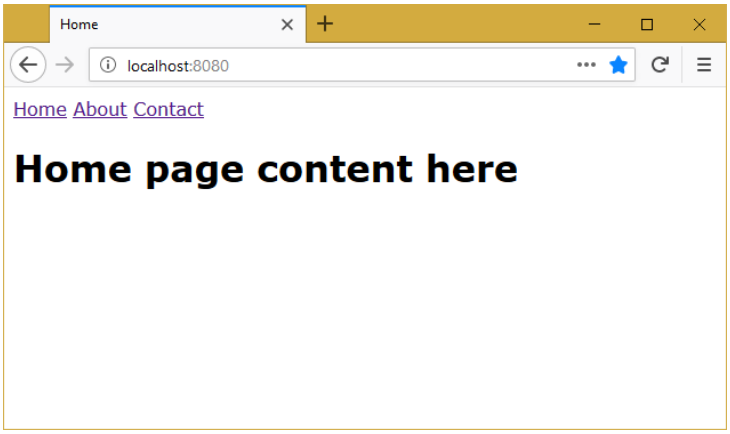
  node httpserver.js

- Open any browser

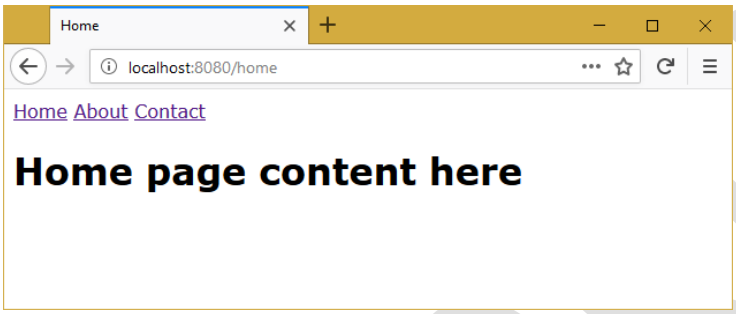
  http://localhost:8080

Output:

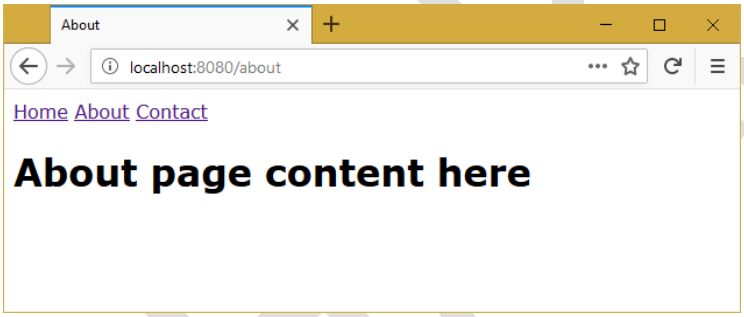




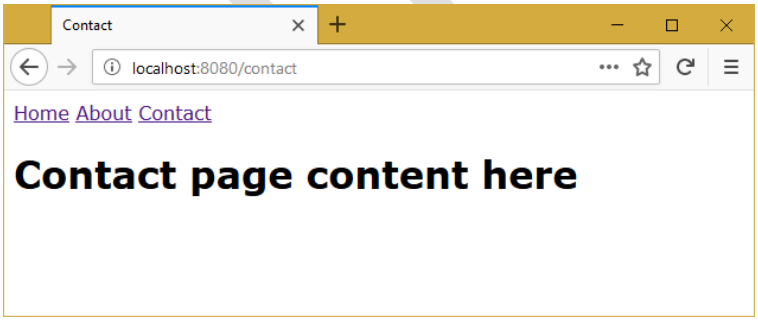
Click on "Home"



Click on "About"



Click on "Contact"



Router

- In "express" package, the Router() function is used to specify the url prefix for a set of routes.

Steps for working with Router

- **Create router:**
  var router = express.Router();

- **Set the URL prefix:**

  app.use("/prefix", router);

- **Receive requests:**
  router.get("url", callbackfunction);
  function callbackfunction(request, response)
  {
  }

- **Send request from browser:**
  http://localhost:portno/url

Router - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "express": "latest"
    }
}
```

c:\nodejs\home.html

```
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        <div id="nav">
            <a href="/admin/home">Home</a>
            <a href="/admin/about">About</a>
            <a href="/admin/contact">Contact</a>
            <a href="/admin/products/mobiles">Mobiles</a>
        </div>
        <h1>Home page content here</h1>
    </body>
</html>
```

c:\nodejs\about.html

```
<html>
    <head>
        <title>About</title>
    </head>
    <body>
        <div id="nav">
            <a href="/admin/home">Home</a>
```

```
            <a href="/admin/about">About</a>
            <a href="/admin/contact">Contact</a>
            <a href="/admin/products/mobiles">Mobiles</a>
        </div>
        <h1>About page content here</h1>
    </body>
</html>
```

c:\nodejs\contact.html

```
<html>
    <head>
        <title>Contact</title>
    </head>
    <body>
        <div id="nav">
            <a href="/admin/home">Home</a>
            <a href="/admin/about">About</a>
            <a href="/admin/contact">Contact</a>
            <a href="/admin/products/mobiles">Mobiles</a>
        </div>
        <h1>Contact page content here</h1>
    </body>
</html>
```

c:\nodejs\404.html

```
<html>
    <head>
        <title>Page not found</title>
    </head>
    <body>
        <h1 style='color:red'>Sorry, page not found</h1>
        <a href="/admin/home">Go to home</a>
    </body>
</html>
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//router
var router = express.Router();
app.use("/admin", router);

//home
router.get("/", fun1);
router.get("/home", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url);
```

```
    response.sendFile(__dirname + "/home.html");
}

//about
router.get("/about", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url);
    response.sendFile(__dirname + "/about.html");
}

//contact
router.get("/contact", fun3);
function fun3(request, response)
{
    console.log("request received at " + request.url);
    response.sendFile(__dirname + "/contact.html");
}

//hello
router.get("/hello", fun4);
function fun4(request, response)
{
    console.log("request received at " + request.url);
    response.write(JSON.stringify(request.query));
    response.end();
}

//products
router.get("/products/:productname", fun5);
function fun5(request, response)
{
    console.log("request received at " + request.url);
    response.header("content-type", "text/html");
    response.status(200);
    console.log(request.params);
    response.write("Product name: " + request.params.productname);
    response.end();
}

//error
app.use(fun6);
function fun6(request, response)
{
    response.sendFile(__dirname + "/404.html");
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

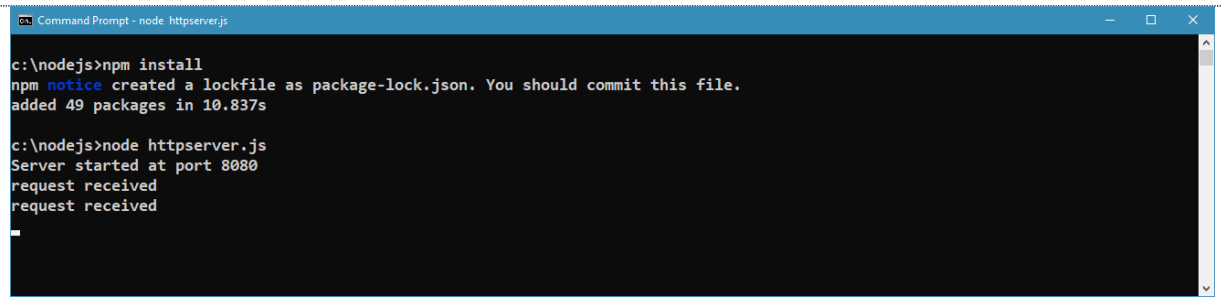
  npm install

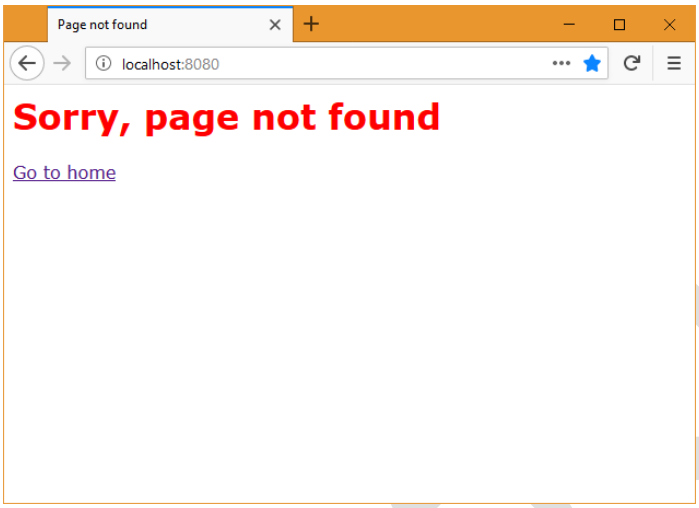
  node httpserver.js

- Open any browser

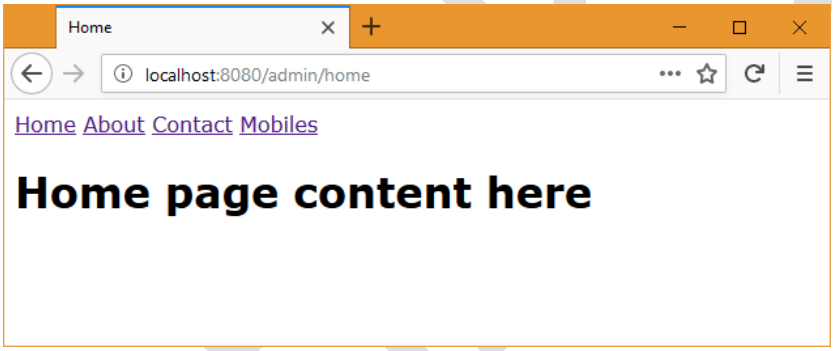
  http://localhost:8080

Output:

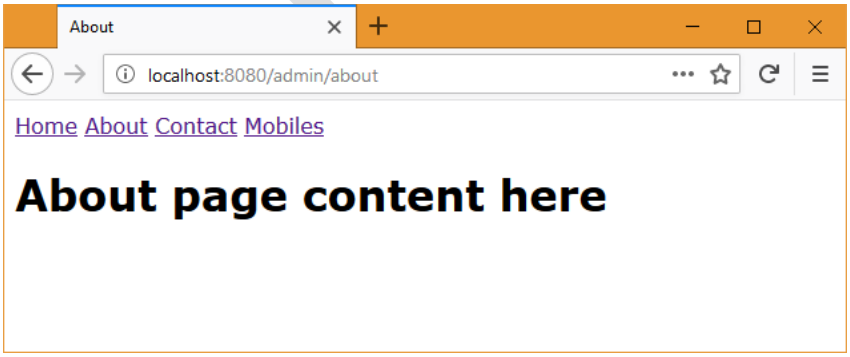




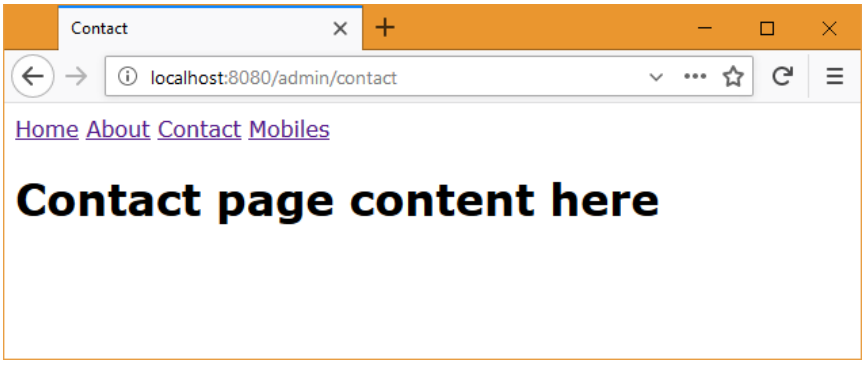
http://localhost:8080/admin/home



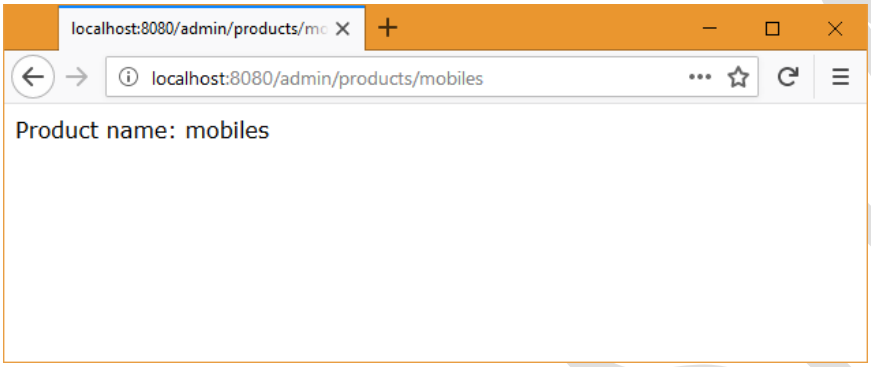
http://localhost:8080/admin/about



http://localhost:8080/admin/contact



http://localhost:8080/admin/products/mobiles



Express - Forms - Get - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "express": "latest"
    }
}
```

c:\nodejs\index.html

```
<html>
    <head>
        <title>Home</title>
    </head>
    <body>
        <form action="/search" method="get">
            Type your search: <input type="text" name="searchbox"><br>
            <input type="submit" value="Search">
        </form>
    </body>
</html>
```

c:\nodejs\404.html

```
<html>
    <head>
        <title>Page not found</title>
    </head>
    <body>
        <h1 style='color:red'>Sorry, page not found</h1>
        <a href="/">Go to home</a>
    </body>
</html>
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//search
app.get("/search", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.query);
    response.write("<h1>Search results for: " + request.query.searchbox + "</h1>");
    response.end();
}

//error
app.use(fun3);
function fun3(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/404.html");
}
```

Execution

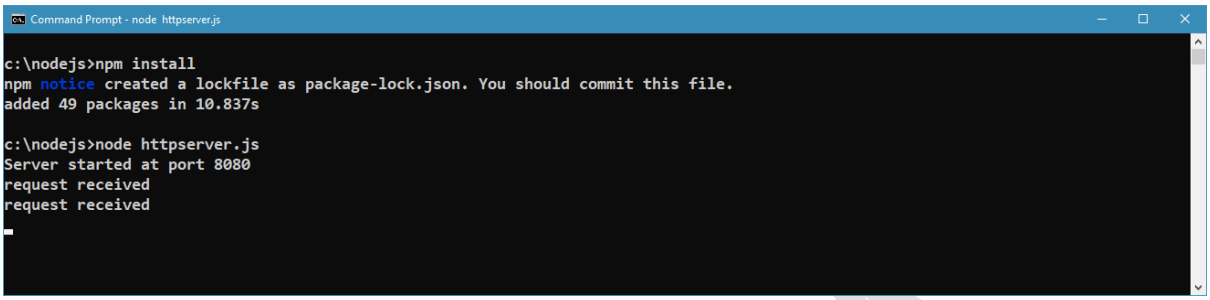
- Open Command Prompt

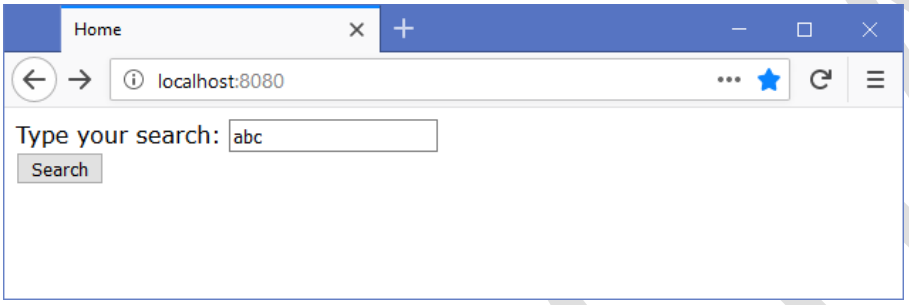
```
cd c:\nodejs
npm install
node httpserver.js
```

- Open any browser

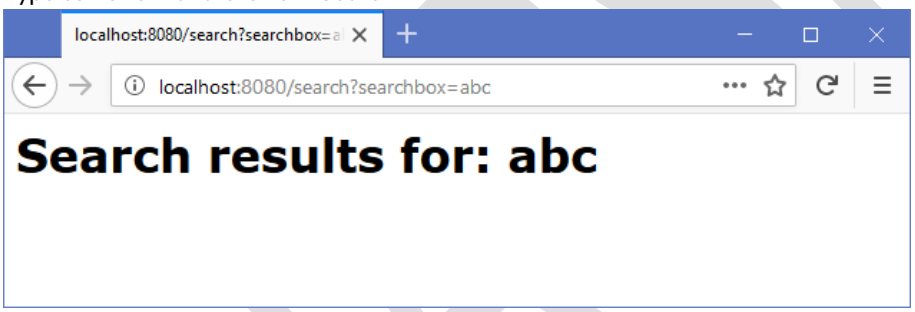
  http://localhost:8080

Output:





Type some text and click on "Search".



"body-parser" package

The "body-parser" package is used to convert the "request body" into "javascript object", in case of "POST" request.

Browser → POST Request → Express → body-parser → Convert post data to js object

- **Steps for working with "body-parser" package:**
  - **Import "body-parser" package in "package.json":**

    ```
    "dependencies":
    {
        "body-parser": "latest"
    }
    ```

  - **Acquire "body-parser" module:**

    ```
    var bodyparser = require("body-parser");
    ```

  - **Receive "application/x-www.form-urlencoded" data:**

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

Note: The default "Content type" of HTTP POST request is: "content-type": "application/x-www-form-urlencoded".

- **Enable "JS Object serialization" in "body parser":**

  app.use(bodyparser.json());

- **Get the request body data of POST request:**

  request.body

Express - Forms - Post - Example

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "body-parser": "latest",
        "express": "latest"
    }
}
```

c:\nodejs\login.html

```
<html>
    <head>
        <title>Login</title>
    </head>
    <body>
        <form action="/login" method="post">
            Username: <input type="text" name="uname"><br>
            Password: <input type="password" name="pwd"><br>
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//bodyparser
var bodyparser = require("body-parser");
app.use(bodyparser.urlencoded({ extended: true }));
app.use(bodyparser.json());
```

```
//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/login.html");
}

//login
app.post("/login", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.body);
    if (request.body.uname == "admin" && request.body.pwd == "server")
    {
        response.write("<h1 style='color:green'>Successful login</h1>");
    }
    else
    {
        response.write("<h1 style='color:red'>Invalid login</h1>");
    }
    response.end();
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

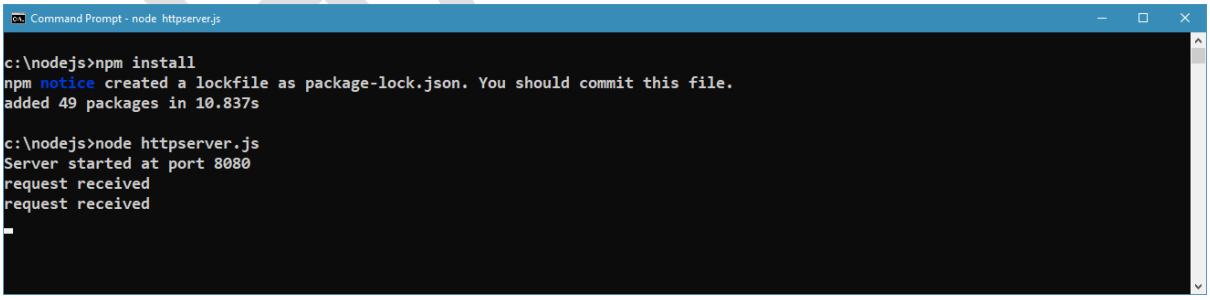
  npm install

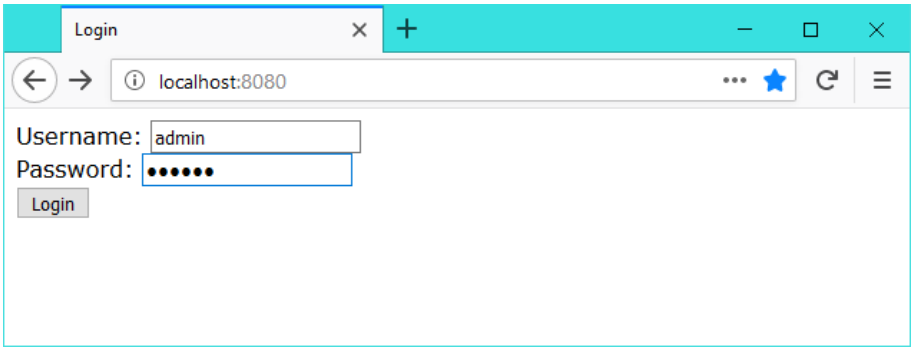
  node httpserver.js

- Open any browser

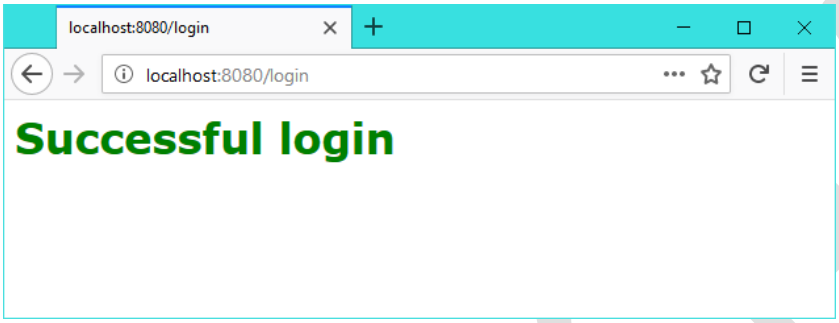
  http://localhost:8080

Output:



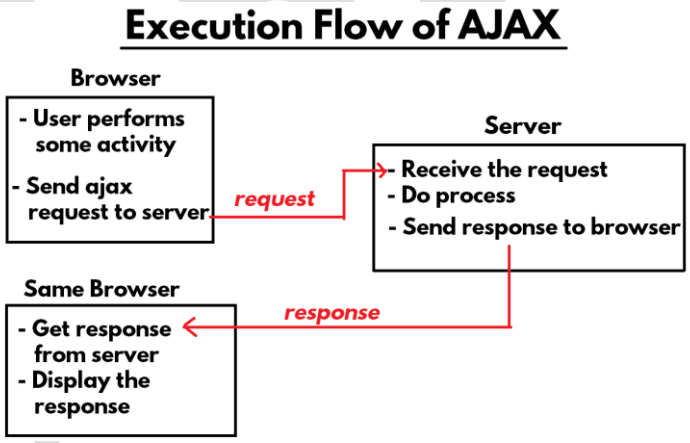


Enter "admin" and "server" as username and password. Click on "Login" button.



AJAX

- AJAX stands for "Asynchronous JavaScript And Xml".
- AJAX is NOT a language, but it is a concept, which is used to send a background request from browser to server and also get background response from server to browser, without refreshing the web page in the browser.
- AJAX allows us to interact with the server and get some data from server, without refreshing the full web page.



- **Advantages of AJAX:**
  1. Quickly interacts with server, without refreshing the web page.
  2. Less data will be exchanged between server and browser, when compared to full page refresh.
  3. User experience is better.
- **Examples:**
  - Facebook like button
  - IRCTC search trains
  - Gmail

etc.

Types of AJAX Request

- In AJAX, the browser can send the following four types of requests:
  1. GET : Used to retrieve / search data from server.
  2. POST : Used to insert data to server.
  3. PUT : Used to update data on server.
  4. DELETE : Used to delete data from server.

JavaScript AJAX

- "JavaScript AJAX" is implemented using a pre-defined class called "XMLHttpRequest".
- This class is supported by all the modern browsers (except IE 6, 7, 8).

Members of XMLHttpRequest

- XMLHttpRequest class
  1. "readyState" property
  2. "onreadystatechange" property
  3. "responseText" property
  4. "open( )" method
  5. "send( )" method

1. open( ) method:

- This method is used to specify the request details such as server url (address), type of the request (GET / POST).
  - **Server url:** The address (url) of server program, to which you want to send request.
  - **Type:** Represents type of the request.

Options: GET | POST | PUT | DELETE

GET : Retrieving data from server
POST : Inserting data
PUT : Updating data
DELETE : Deleting data

  - **Syntax:** open("type of request", "url")
  - **Example:** open("get", "http://localhost:7070")

2. send( ) method:

- This method sends an asynchronous request (background request) to server.
  - **Syntax:** send( )
  - **Example:** send( )

3. readyState

- It represents a number, that represents current status of the request.
  - **Syntax:** readyState
  - **Example:** readyState

0 : Request not initialized.

1	:	Request opened (open( ) method is called).
2	:	Request sent to server (send( ) method called).
3	:	Server started processing the request.
4	:	Response received from server.

4. **onreadystatechange:**
   - This property stores a callback function, which executes automatically, when the "readyState" property gets changed.
     - **Syntax:** onreadystatechange = functionname;
     - **Example:** onreadystatechange = fun1;

5. **responseText:**
   - This property stores the actual response sent from the server to the browser.
   - It represents the data in string format.
     - **Syntax:** responseText
     - **Example:** responseText

JavaScript AJAX - NodeJS - Simple - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "express": "*"
    }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}
app.use(express.static(__dirname));
```

```
//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//getdata
app.get("/getdata", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.send("Hello from server at " + new Date().toLocaleTimeString());
}
```

c:\nodejs\index.html

```
<html>
    <head>
        <title>AJAX - First Example</title>
    </head>
    <body>
        <h1>AJAX - First Example</h1>
        <input type="button" id="button1" value="GET data from server">
        <div id="div1"></div>

        <script>
            var xhr;
            document.getElementById("button1").addEventListener("click", fun1);
            function fun1()
            {
                xhr = new XMLHttpRequest();
                xhr.open("GET", "/getdata");
                xhr.onreadystatechange = fun2;
                xhr.send();
            }

            function fun2()
            {
                if (xhr.readyState == 4)
                {
                    document.getElementById("div1").innerHTML += xhr.responseText + "<br>";
                }
            }
        </script>
    </body>
</html>
```

Execution:

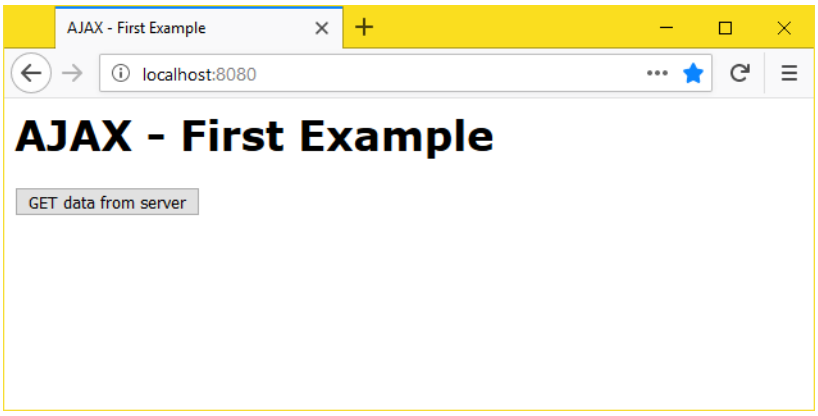
- Open Command Prompt

  cd c:\nodejs

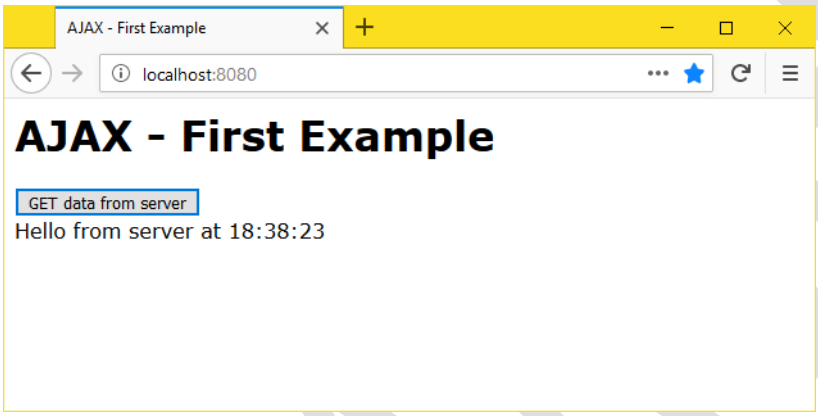
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Click on "GET data from server".



JavaScript AJAX - NodeJS - Get - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
    "express": "*",
    "mongoose": "*"
```

```
    }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//mongoose
var mongoose = require("mongoose");
var EmployeesSchema  = new mongoose.Schema({ empid: Number, empname: String, salary: Number });
var Employee = mongoose.model("employees", EmployeesSchema);

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//getemployees
app.get("/getemployees", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    mongoose.connect("mongodb://localhost/company");
    Employee.find(fun3);
    function fun3(error, data)
    {
        if (error)
        {
            console.log(error);
            response.end();
        }
        else
        {
            response.header("content-type", "application/json");
            response.send(data);
            mongoose.connection.close();
        }
    }
}
```

c:\nodejs\index.html

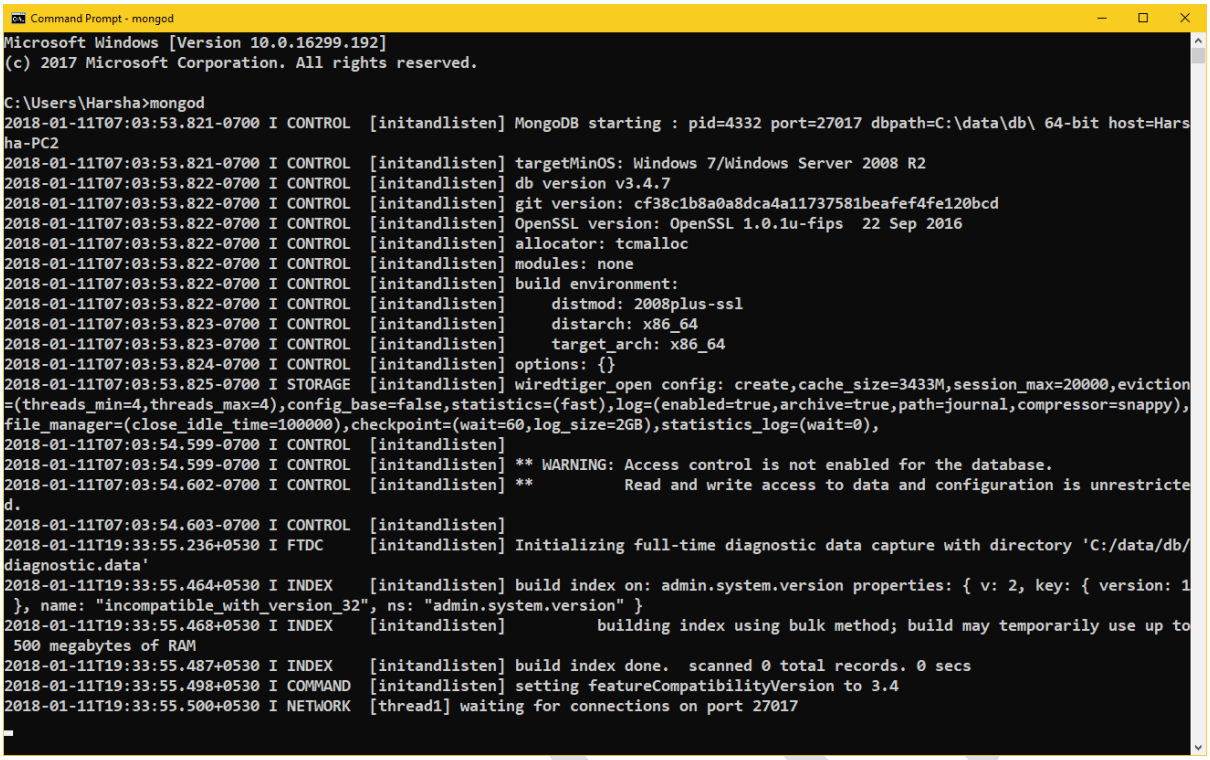
```
<html>
    <head>
        <title>AJAX - GET</title>
```

```
    </head>
    <body>
        <h1>AJAX - GET</h1>
        <input type="button" id="button1" value="GET data from server">
        <table id="table1" border="1">
        </table>

        <script>
            var xhr;
            document.getElementById("button1").addEventListener("click", fun1);
            function fun1()
            {
                xhr = new XMLHttpRequest();
                xhr.open("GET", "/getemployees");
                xhr.onreadystatechange = fun2;
                xhr.send();
            }

            function fun2()
            {
                if (xhr.readyState == 4)
                {
                    var emps = JSON.parse(xhr.responseText); //string to json
                    document.getElementById("table1").innerHTML = "<tr> <th>Emp ID</th> <th>Emp Name</th>
<th>Salary</th> </tr>";
                    for (var i = 0; i < emps.length; i++)
                    {
                        document.getElementById("table1").innerHTML += "<tr> <td>" + emps[i].empid + "</td> <td>" +
emps[i].empname + "</td> <td>" + emps[i].salary + "</td> </tr>";
                    }
                }
            }

        </script>
    </body>
</html>
```

Execution:

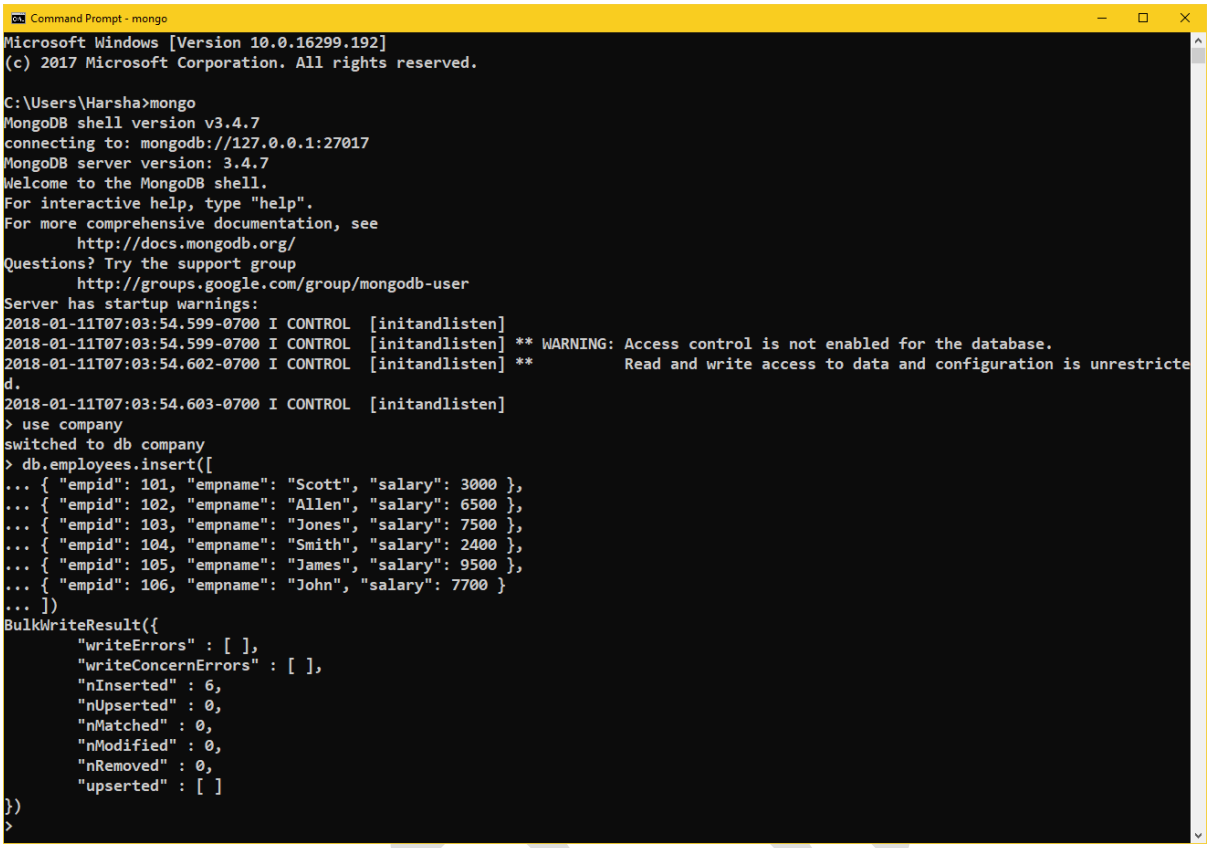
- Open Command Prompt 1

  mongod



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```



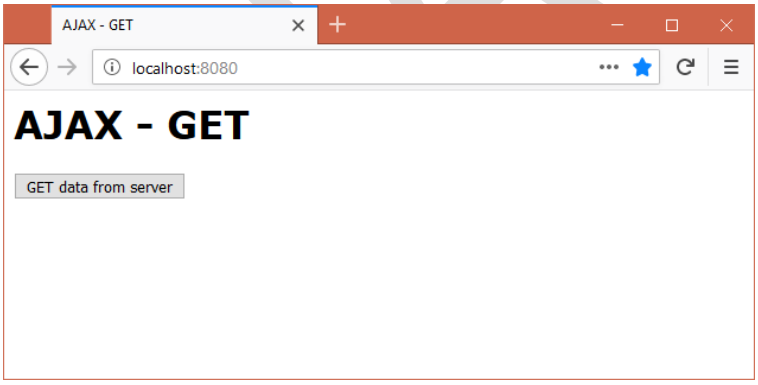
- Open Command Prompt 3

  cd c:\nodejs

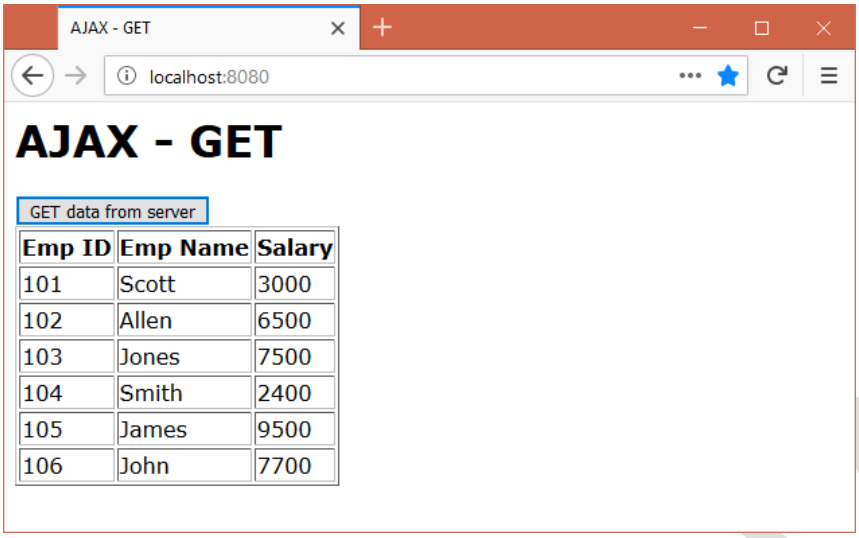
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Click on "GET data from server".



JavaScript AJAX - NodeJS - Search - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "express": "*",
      "mongoose": "*"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
}
app.use(express.static(__dirname));
```

```
//mongoose
var mongoose = require("mongoose");
var EmployeesSchema  = new mongoose.Schema({ empid: Number, empname: String, salary: Number });
var Employee = mongoose.model("employees", EmployeesSchema);

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//searchemployees
app.get("/searchemployees", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    mongoose.connect("mongodb://localhost/company");
    Employee.find({ "empname": { $regex: request.query.str } }, fun3);
    function fun3(error, data)
    {
        if (error)
        {
            console.log(error);
            mongoose.connection.close();
            response.end();
        }
        else
        {
            response.header("content-type", "application/json");
            response.send(data);
            mongoose.connection.close();
        }
    }
}
```

c:\nodejs\index.html

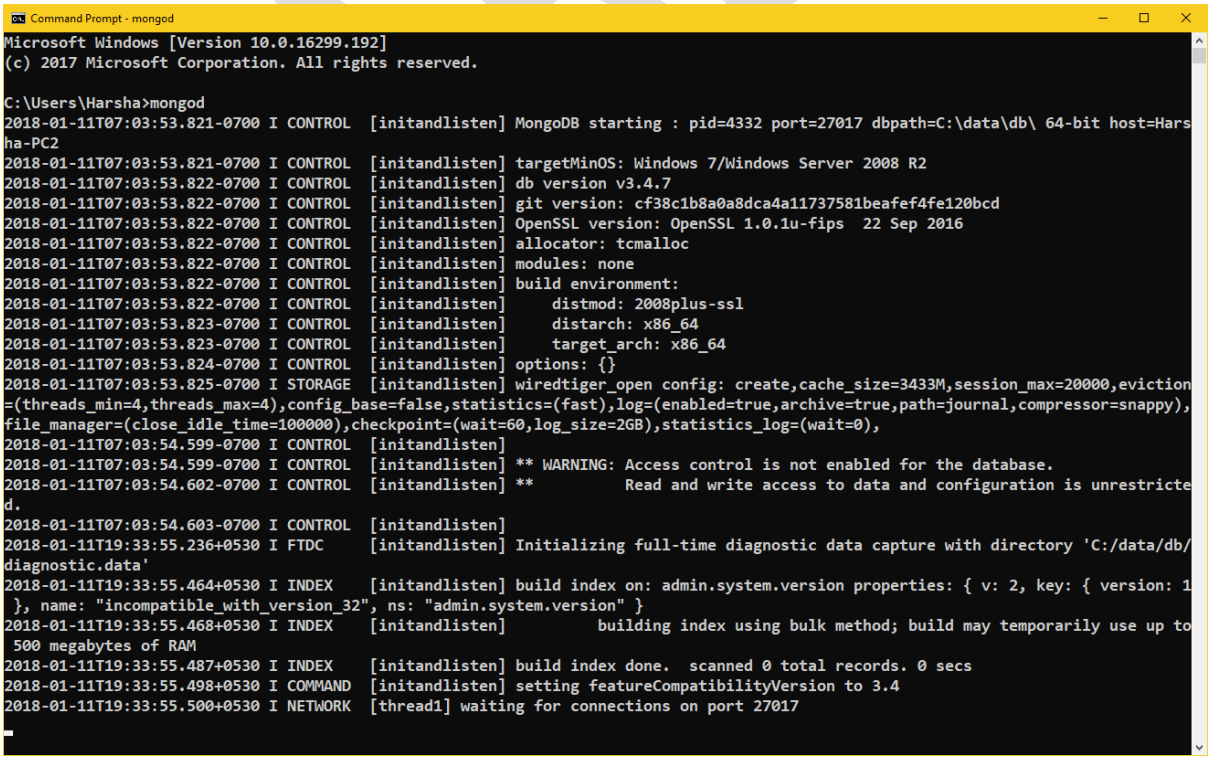
```
<html>
    <head>
        <title>AJAX - Search</title>
    </head>
    <body>
        <h1>AJAX - Search</h1>
        Search: <input type="text" id="txt1"><br>
        <input type="button" id="button1" value="Search">
        <table id="table1" border="1">
        </table>

        <script>
            var xhr;
            document.getElementById("button1").addEventListener("click", fun1);

            function fun1()
            {
```

```
            var s = document.getElementById("txt1").value;
            xhr = new XMLHttpRequest();
            xhr.open("GET", "/searchemployees?str=" + s);
            xhr.onreadystatechange = fun2;
            xhr.send();
        }

        function fun2()
        {
            if (xhr.readyState == 4)
            {
                var emps = JSON.parse(xhr.responseText); //string to json
                document.getElementById("table1").innerHTML = "<tr> <th>Emp ID</th> <th>Emp Name</th>
<th>Salary</th> </tr>";
                for (var i = 0; i < emps.length; i++)
                {
                    document.getElementById("table1").innerHTML += "<tr> <td>" + emps[i].empid + "</td> <td>" +
emps[i].empname + "</td> <td>" + emps[i].salary + "</td> </tr>";
                }
            }
        }
    </script>
  </body>
</html>
```

Execution:

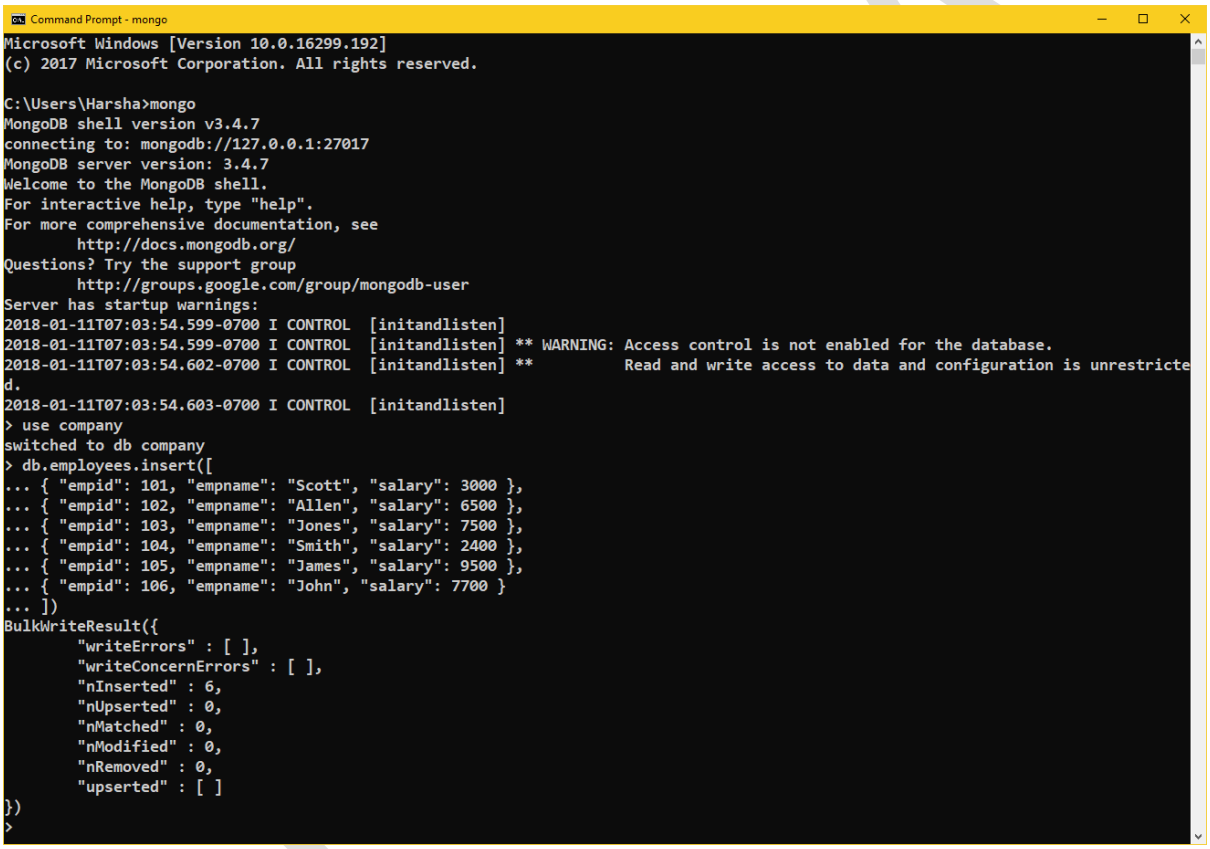
- Open Command Prompt 1

  mongod



- Open Command Prompt 2

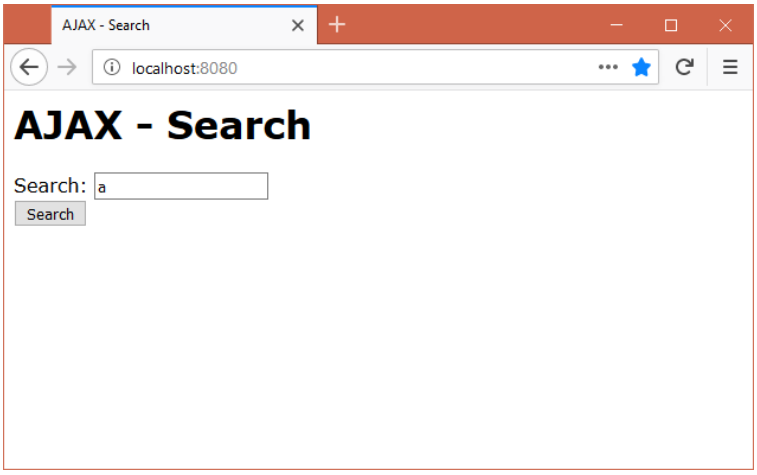
```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```



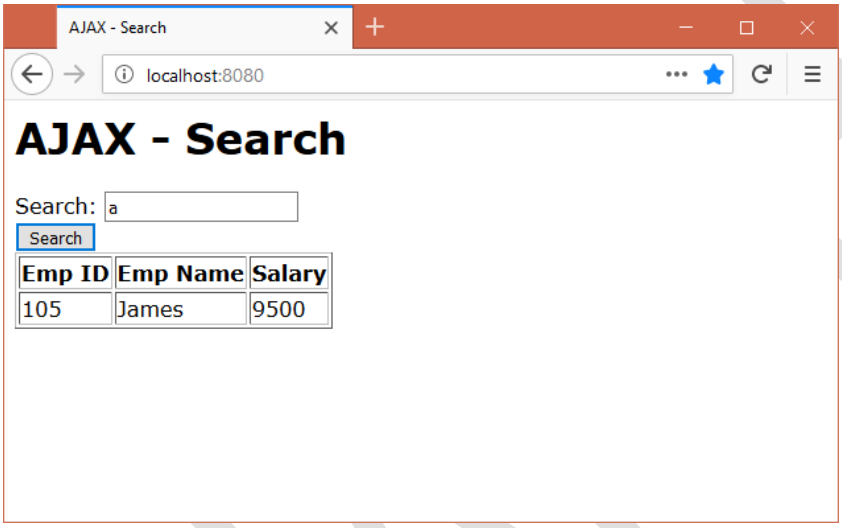
- Open Command Prompt 3

```
cd c:\nodejs
npm install
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080



Enter some text and click on "Search".



JavaScript AJAX - NodeJS - Post - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
```

```
    {
        "express": "*",
        "mongoose": "*",
        "body-parser": "*"
    }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//mongoose
var mongoose = require("mongoose");
var EmployeeSchema = new mongoose.Schema({ empid: Number, empname: String, salary: Number }, { versionKey:
false });
var Employee = mongoose.model("employees", EmployeeSchema);

//bodyparser
var bodyparser = require("body-parser");
app.use(bodyparser.json());
//app.use(bodyparser.text());
app.use(bodyparser.urlencoded({ extended: true }));

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//insertemployee
app.post("/insertemployee", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.body);
    mongoose.connect("mongodb://localhost/company");
    var newemp = new Employee({ empid: request.body.empid, empname: request.body.empname, salary:
request.body.salary});
    newemp.save(fun3);
    function fun3(error)
    {
        if (error)
        {
            console.log(error);
            response.end();
```

```
        }
        else
        {
            response.send("Successfully inserted");
            mongoose.connection.close();
        }
    }
}
```

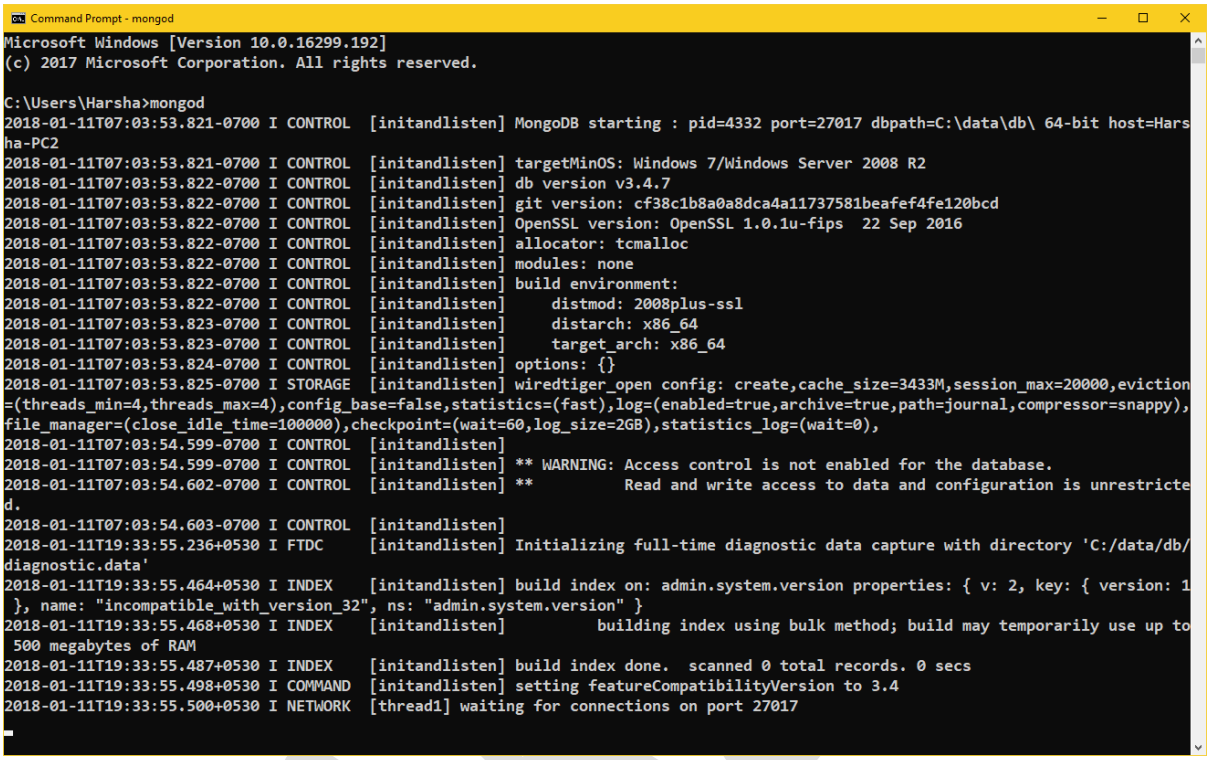
c:\nodejs\index.html

```
<html>
    <head>
        <title>AJAX - POST</title>
    </head>
    <body>
        <h1>AJAX - POST</h1>
        <form>
            Emp ID: <input type="text" id="txt1"><br>
            Emp Name: <input type="text" id="txt2"><br>
            Salary: <input type="text" id="txt3"><br>
            <input type="submit" value="Insert" id="button1">
            <div id="div1"></div>
        </form>

        <script>
            var xhr;
            document.getElementById("button1").addEventListener("click", fun1);

            function fun1(event)
            {
                event.preventDefault();
                var a = document.getElementById("txt1").value;
                var b = document.getElementById("txt2").value;
                var c = document.getElementById("txt3").value;
                var emp = {"empid": a, "empname": b, "salary": c};
                var s = JSON.stringify(emp);

                xhr = new XMLHttpRequest();
                xhr.open("POST", "/insertemployee");
                xhr.setRequestHeader("Content-Type", "application/json");
                xhr.onreadystatechange = fun2;
                xhr.send(s);
            }

            function fun2()
            {
                if (xhr.readyState == 4)
                {
                    document.getElementById("div1").innerHTML = xhr.responseText;
                }
            }
        </script>
    </body>
</html>
```

Execution:

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

  mongo

  use company

  db.employees.insert([

  { "empid": 101, "empname": "Scott", "salary": 3000 },

  { "empid": 102, "empname": "Allen", "salary": 6500 },

  { "empid": 103, "empname": "Jones", "salary": 7500 },

  { "empid": 104, "empname": "Smith", "salary": 2400 },

  { "empid": 105, "empname": "James", "salary": 9500 },

  { "empid": 106, "empname": "John", "salary": 7700 }

  ])

```
Command Prompt - mongo
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>mongo
MongoDB shell version v3.4.7
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.7
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten]
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-01-11T07:03:54.602-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricte
d.
2018-01-11T07:03:54.603-0700 I CONTROL  [initandlisten]
> use company
switched to db company
> db.employees.insert([
... { "empid": 101, "empname": "Scott", "salary": 3000 },
... { "empid": 102, "empname": "Allen", "salary": 6500 },
... { "empid": 103, "empname": "Jones", "salary": 7500 },
... { "empid": 104, "empname": "Smith", "salary": 2400 },
... { "empid": 105, "empname": "James", "salary": 9500 },
... { "empid": 106, "empname": "John", "salary": 7700 }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

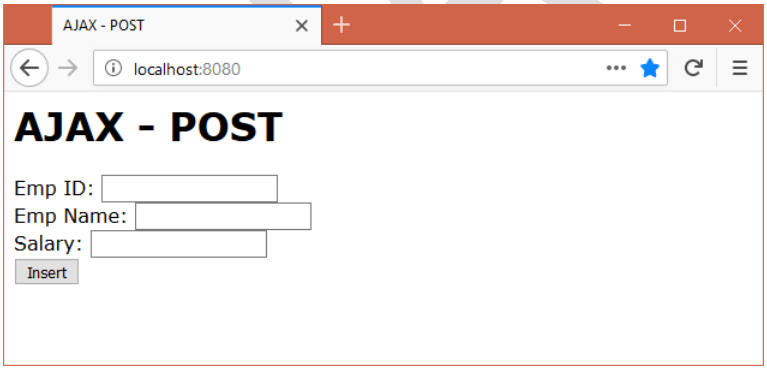
- Open Command Prompt 3

  cd c:\nodejs

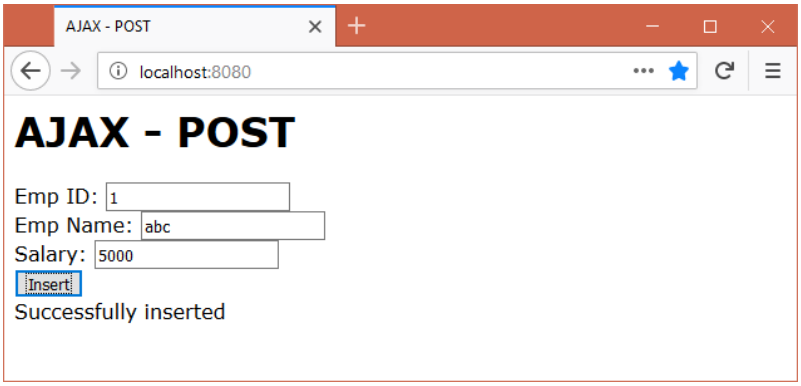
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Enter some details and click on "Insert".



JavaScript AJAX - NodeJS - Put - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
     "express": "*",
     "mongoose": "*",
     "body-parser": "*"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//mongoose
var mongoose = require("mongoose");
```

```
var EmployeeSchema = new mongoose.Schema({ "empid": Number, "empname": String, "salary": Number }, {
versionKey: false });
var Employee = mongoose.model("employees", EmployeeSchema);

//bodyparser
var bodyparser = require("body-parser");
app.use(bodyparser.json());
app.use(bodyparser.urlencoded({ extended: true }));

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//updateemployee
app.put("/updateemployee", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.body);
    mongoose.connect("mongodb://localhost/company");
    Employee.findOne({ empid: request.body.empid }, fun3);
    function fun3(err, data)
    {
        if(err)
        {
            response.send("Error");
        }
        else
        {
            data.empname = request.body.empname;
            data.salary = request.body.salary;
            data.save(fun4);
            function fun4(error)
            {
                if(error)
                {
                    response.send("Not updated");
                }
                else
                {
                    response.send("Successfully Updated");
                }
                mongoose.connection.close();
            }
        }
    }
}
```

c:\nodejs\index.html

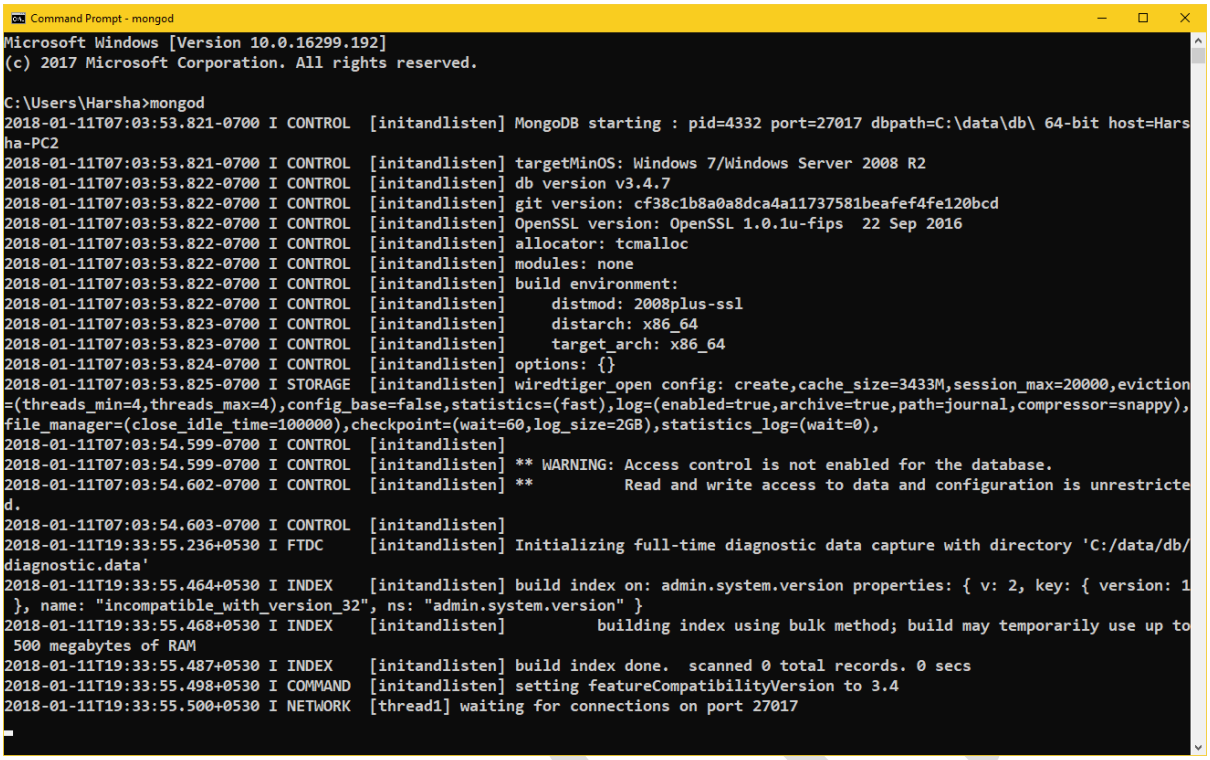
```
<html>
    <head>
```

```
        <title>AJAX - PUT</title>
    </head>
    <body>
        <h1>AJAX - PUT</h1>
        <form>
            Existing Emp ID: <input type="text" id="txt1"><br>
            Emp Name: <input type="text" id="txt2"><br>
            Salary: <input type="text" id="txt3"><br>
            <input type="submit" value="Update" id="button1">
            <div id="div1"></div>
        </form>

        <script>
            var xhr;
            document.getElementById("button1").addEventListener("click", fun1);

            function fun1(event)
            {
                event.preventDefault();
                var a = document.getElementById("txt1").value;
                var b = document.getElementById("txt2").value;
                var c = document.getElementById("txt3").value;
                var emp = { "empid": a, "empname": b, "salary": c };
                var s = JSON.stringify(emp);

                xhr = new XMLHttpRequest();
                xhr.open("PUT", "/updateemployee");
                xhr.setRequestHeader("Content-Type", "application/json");
                xhr.onreadystatechange = fun2;
                xhr.send(s);
            }

            function fun2()
            {
                if (xhr.readyState == 4)
                {
                    document.getElementById("div1").innerHTML = xhr.responseText;
                }
            }
        </script>
    </body>
</html>
```

Execution:

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```

```
Command Prompt - mongo
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>mongo
MongoDB shell version v3.4.7
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.7
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten]
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-01-11T07:03:54.602-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricte
d.
2018-01-11T07:03:54.603-0700 I CONTROL  [initandlisten]
> use company
switched to db company
> db.employees.insert([
... { "empid": 101, "empname": "Scott", "salary": 3000 },
... { "empid": 102, "empname": "Allen", "salary": 6500 },
... { "empid": 103, "empname": "Jones", "salary": 7500 },
... { "empid": 104, "empname": "Smith", "salary": 2400 },
... { "empid": 105, "empname": "James", "salary": 9500 },
... { "empid": 106, "empname": "John", "salary": 7700 }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

- Open Command Prompt 3

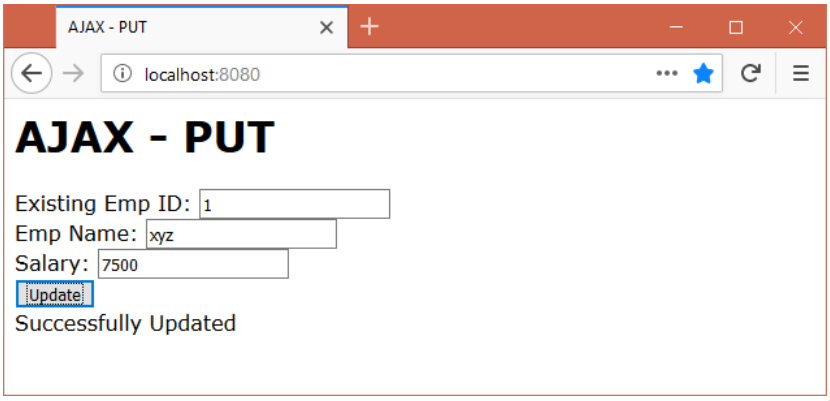
  cd c:\nodejs

  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080

Enter some details and click on "Update".



JavaScript AJAX - NodeJS - Delete - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
     "express": "*",
     "mongoose": "*"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//mongoose
var mongoose = require("mongoose");
var EmployeeSchema = new mongoose.Schema({ "empid": Number, "empname": String, "salary": Number }, {
versionKey: false });
```

```
var Employee = mongoose.model("employees", EmployeeSchema);

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//deleteemployee
app.delete("/deleteemployee", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    mongoose.connect("mongodb://localhost/company");
    Employee.remove({ empid: request.query.empid }, fun3);
    function fun3(err)
    {
        if(err)
        {
            response.send("Error");
        }
        else
        {
            response.send("Successfully Deleted");
            mongoose.connection.close();
        }
    }
}
```

c:\nodejs\index.html

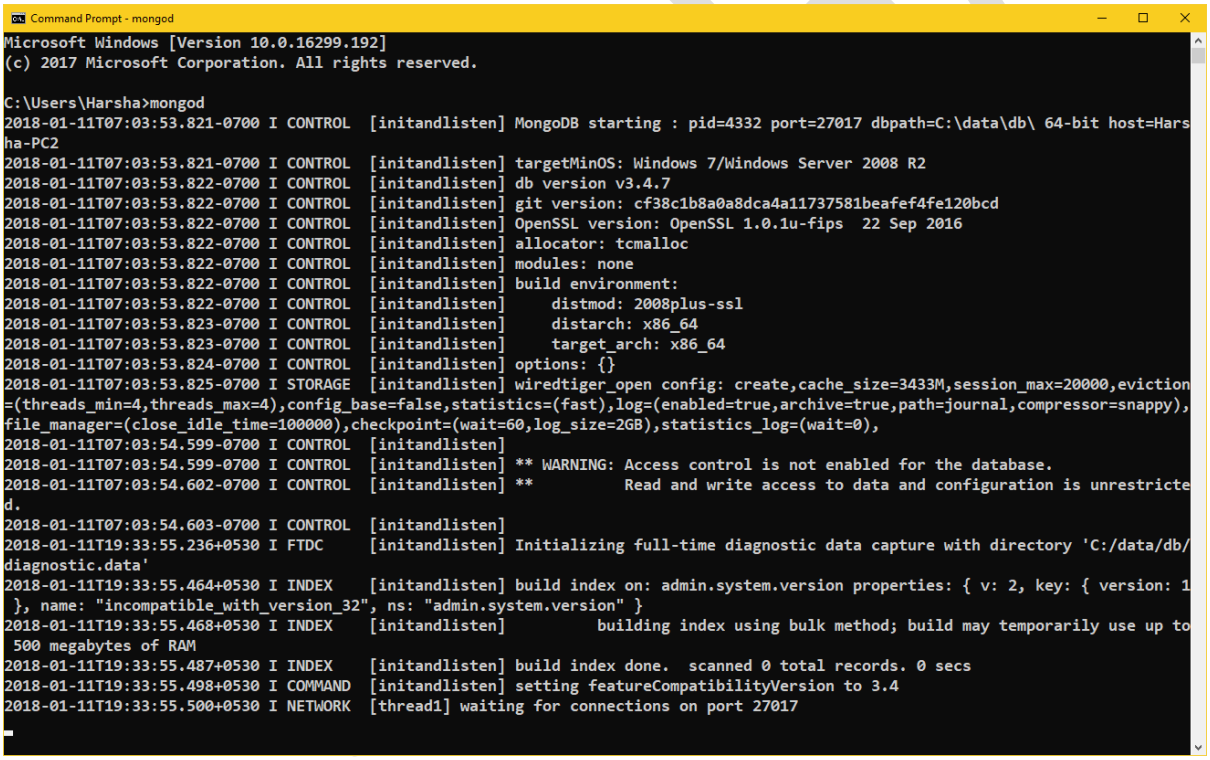
```
<html>
    <head>
        <title>AJAX - DELETE</title>
    </head>
    <body>
        <h1>AJAX - DELETE</h1>
        <form>
            Existing Emp ID: <input type="text" id="txt1"><br>
            <input type="submit" value="Delete" id="button1">
            <div id="div1"></div>
        </form>

        <script>
            var xhr;
            document.getElementById("button1").addEventListener("click", fun1);

            function fun1(event)
            {
                event.preventDefault();
                var a = document.getElementById("txt1").value;

                xhr = new XMLHttpRequest();
                xhr.open("DELETE", "/deleteemployee?empid=" + a);
```

```
            xhr.onreadystatechange = fun2;
            xhr.send();
        }

        function fun2()
        {
            if (xhr.readyState == 4)
            {
                document.getElementById("div1").innerHTML = xhr.responseText;
            }
        }
    </script>
  </body>
</html>
```

Execution:

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

  mongo

  use company

  db.employees.insert([

  { "empid": 101, "empname": "Scott", "salary": 3000 },

  { "empid": 102, "empname": "Allen", "salary": 6500 },

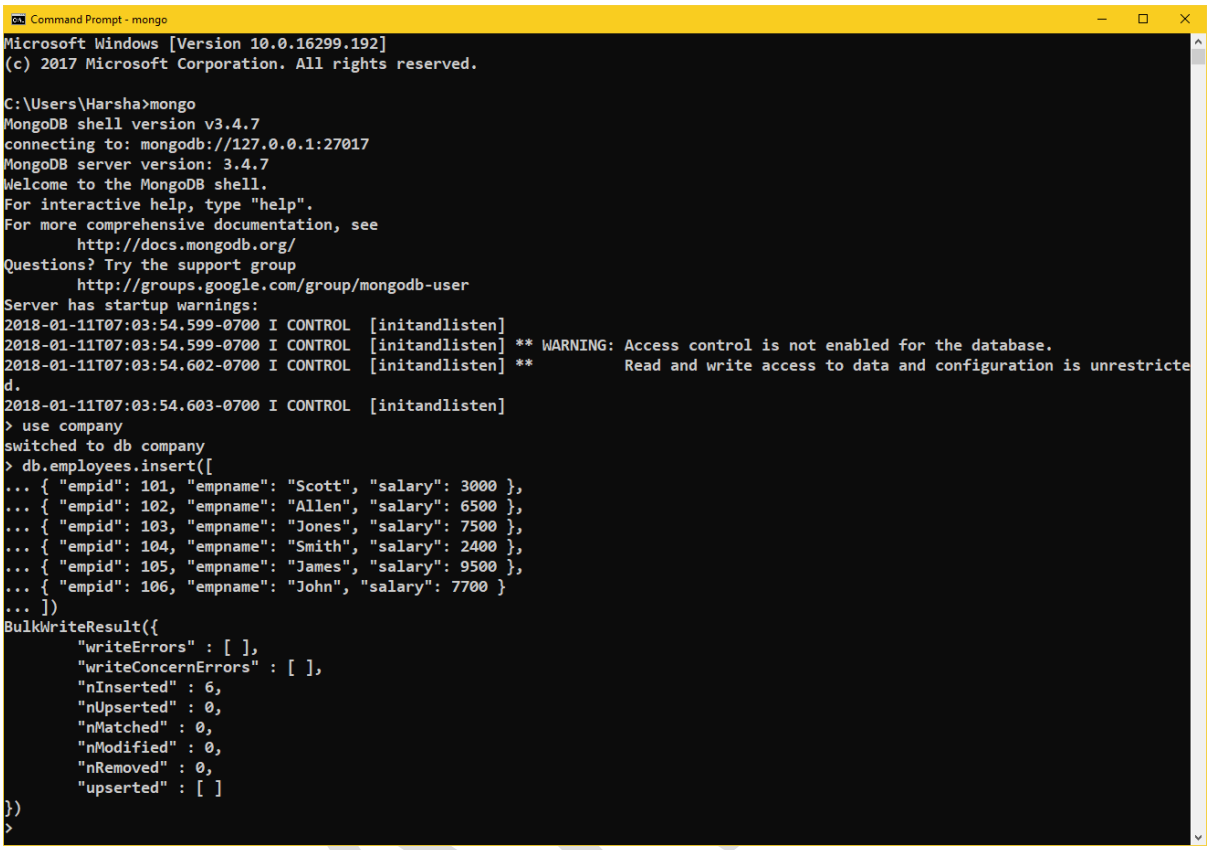
  { "empid": 103, "empname": "Jones", "salary": 7500 },

{ "empid": 104, "empname": "Smith", "salary": 2400 },

{ "empid": 105, "empname": "James", "salary": 9500 },

{ "empid": 106, "empname": "John", "salary": 7700 }

])



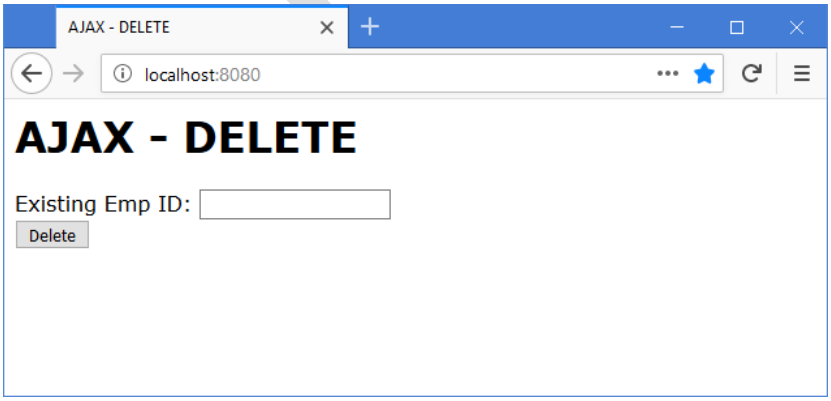
- Open Command Prompt 3

  cd c:\nodejs

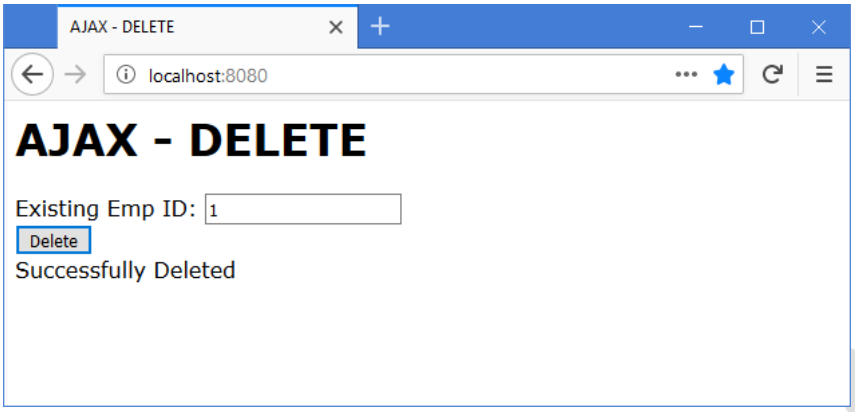
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Enter some emp id and click on "Delete".



jQuery AJAX

- "jQuery" provides a method called "$.ajax()", to send ajax request to server and get ajax response from server easily.

Syntax of $.ajax()

```
$.ajax( { type: "GET | POST | PUT | DELETE", url: "address", success: callbackfunction, error: callbackfunction } )
```

1. **url**
   - Represents the server address, to which you want to send request.
2. **type**
   - Represents the server address, to which you want to send request.
     - **GET** : Retrieving data from server
     - **POST** : Inserting data
     - **PUT** : Updating data
     - **DELETE** : Deleting data
3. **success**
   - Represents the callback function, which will be called automatically after successfully receiving the response from the server.
4. **error**
   - Represents the callback function, which will be called automatically after successfully receiving the response from the server.

jQuery AJAX – NodeJS – Simple - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
```

```
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
   "jquery": "latest",
   "express": "latest"
  }
 }
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//getdata
app.get("/getdata", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.send("Hello from server at " + new Date().toLocaleTimeString());
}
```

c:\nodejs\index.html

```
<html>
    <head>
        <title>jQuery AJAX - Simple</title>
    </head>
   <body>
    <h1>jQuery AJAX - Simple</h1>
    <input type="button" id="button1" value="GET data from server">
    <div id="div1"></div>

    <script src="node_modules/jquery/dist/jquery.js"></script>

    <script>
      $("#button1").click(fun1);
      function fun1(event)
      {
```

```
    event.preventDefault();
    $.ajax({ type: "GET", url: "/getdata", success: fun2, error: fun3 });
  }

  function fun2(response)
  {
    $("#div1").html(response);
  }

  function fun3(error)
  {
    alert(error);
  }
 </script>
</body>
</html>
```

Execution:

- Open Command Prompt

  cd c:\nodejs

  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080

Click on "GET data from server".

jQuery AJAX - NodeJS - Get - Example

Creating the application

- Create "c:\nodejs" folder.

- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "jquery": "latest",
      "express": "latest",
      "mongoose": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//mongoose
var mongoose = require("mongoose");
var EmployeesSchema  = new mongoose.Schema({ empid: Number, empname: String, salary: Number });
var Employee = mongoose.model("employees", EmployeesSchema);

//home
app.get("/", fun1);
function fun1(request, response)
{
   console.log("request received at " + request.url + ", " + request.method);
   response.sendFile(__dirname + "/index.html");
}

//getemployees
app.get("/getemployees", fun2);
function fun2(request, response)
{
   console.log("request received at " + request.url + ", " + request.method);
   mongoose.connect("mongodb://localhost/company");

   Employee.find(fun3);
```

```
    function fun3(error, data)
    {
        if (error)
        {
            console.log(error);
            response.end();
        }
        else
        {
            console.log(data);
            response.header("content-type", "application/json");
            response.send(data);
            mongoose.connection.close();
        }
    }
}
```

c:\nodejs\index.html

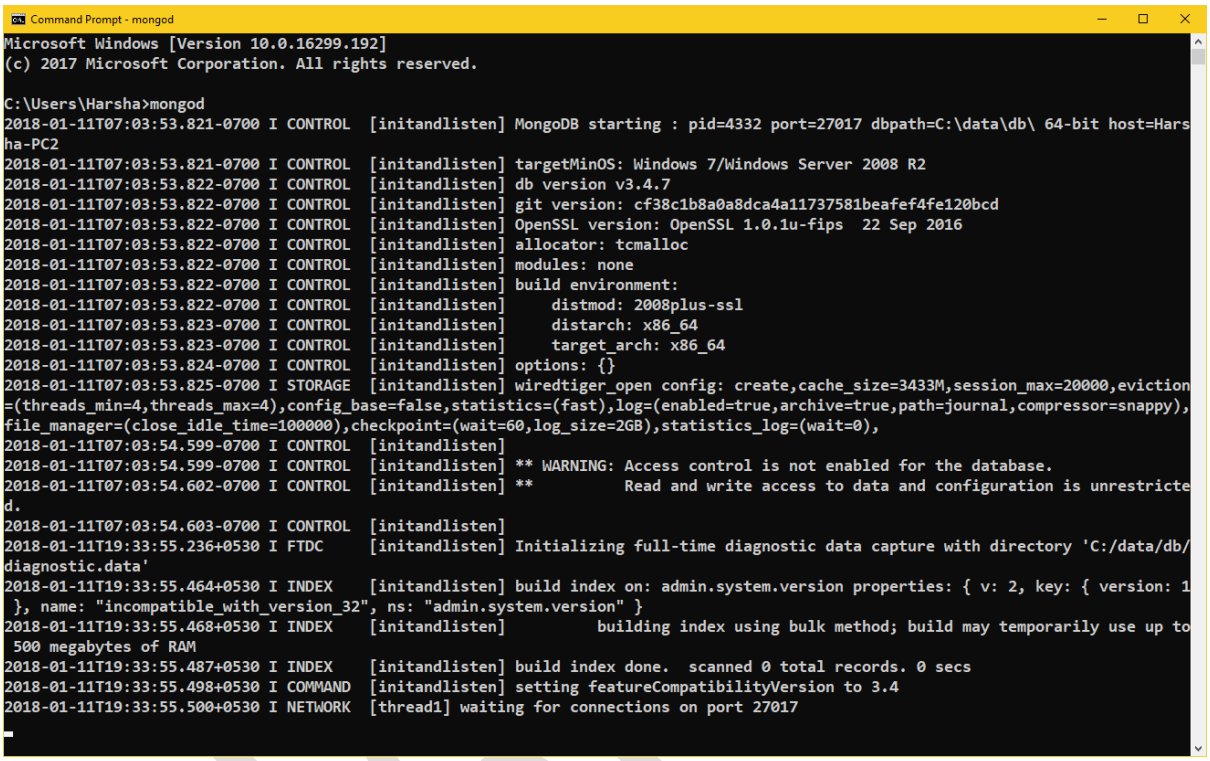
```
<!DOCTYPE html>
<html>
<head>
  <title>NodeJS - jQuery AJAX - Get</title>
</head>
<body>
  <h1>NodeJS - jQuery AJAX - Get</h1>
  <form>
    <input type="button" id="btn1" value="Get Data">
    <table id="table1" border="1">
      <tr> <th>Emp ID</th> <th>Emp Name</th> <th>Salary</th> </tr>

    </table>
  </form>

  <script src="node_modules/jquery/dist/jquery.js"></script>

  <script>
    $("#btn1").click(fun1);
    function fun1(event)
    {
      event.preventDefault();
      $.ajax({ type: "GET", url: "/getemployees", success: fun2, error: fun3 });
    }

    function fun2(response)
    {
      $("#table1 tr:gt(0)").remove();
      for (var i = 0; i < response.length; i++)
      {
        $("#table1").append("<tr> <td>" + response[i].empid + "</td> <td>" + response[i].empname + "</td> <td>" +
response[i].salary + "</td> </tr>");
      }
    }

    function fun3(error)
    {
```

```
        alert(error);
      }
    </script>
  </body>
  </html>
```

Execution:

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

  mongo

  use company

  db.employees.insert([

  { "empid": 101, "empname": "Scott", "salary": 3000 },

  { "empid": 102, "empname": "Allen", "salary": 6500 },

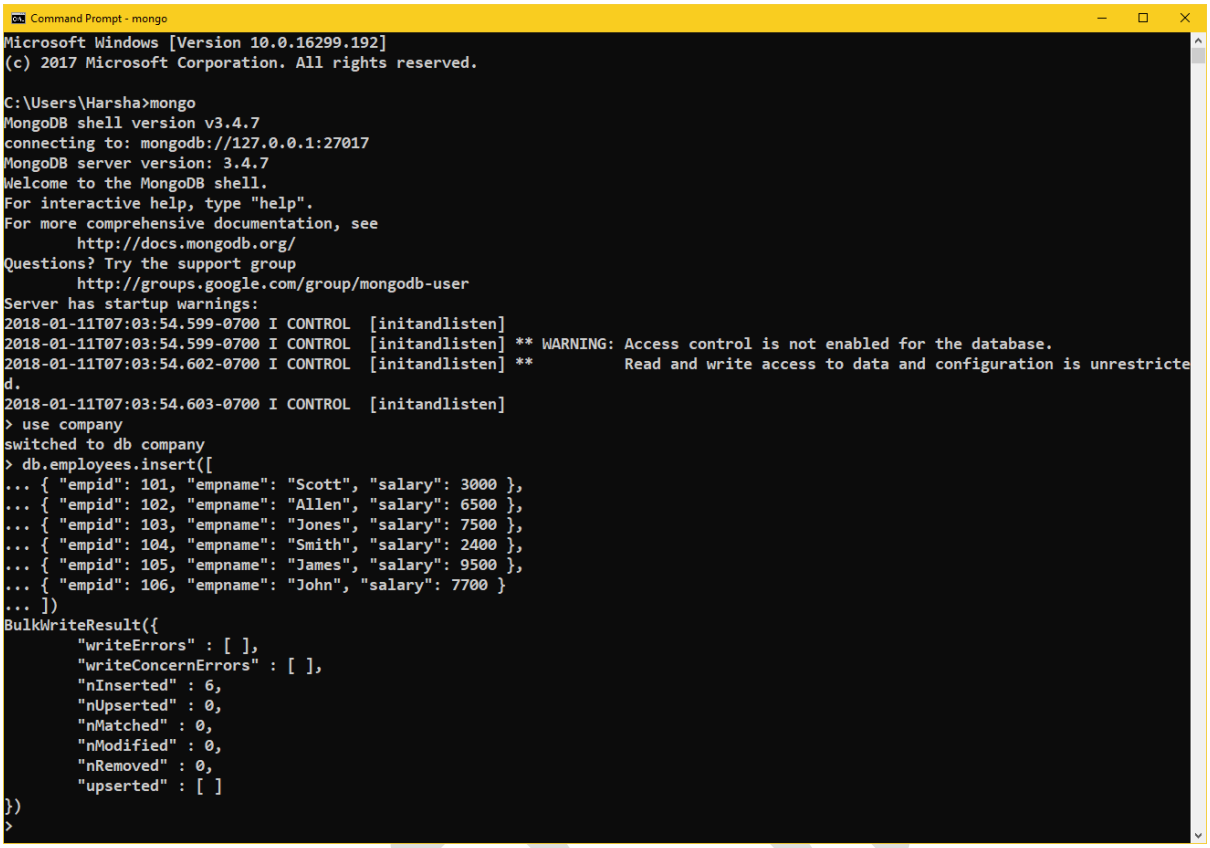
  { "empid": 103, "empname": "Jones", "salary": 7500 },

  { "empid": 104, "empname": "Smith", "salary": 2400 },

  { "empid": 105, "empname": "James", "salary": 9500 },

  { "empid": 106, "empname": "John", "salary": 7700 }

  ])



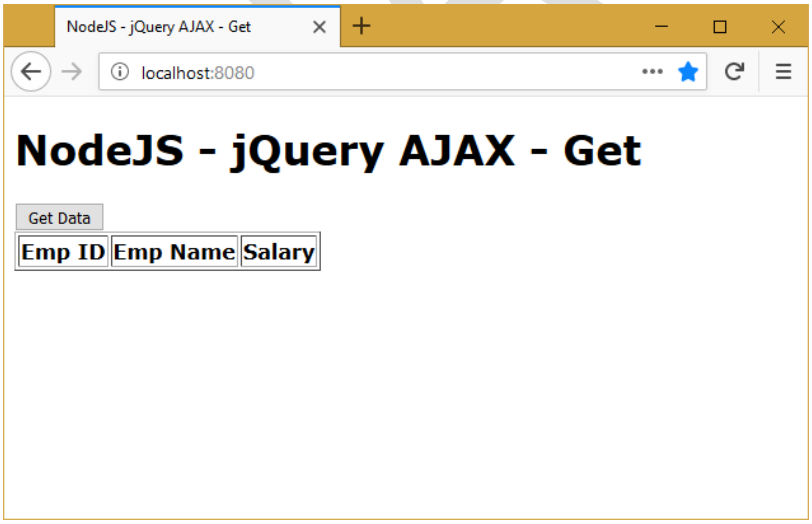
- Open Command Prompt 3

  cd c:\nodejs

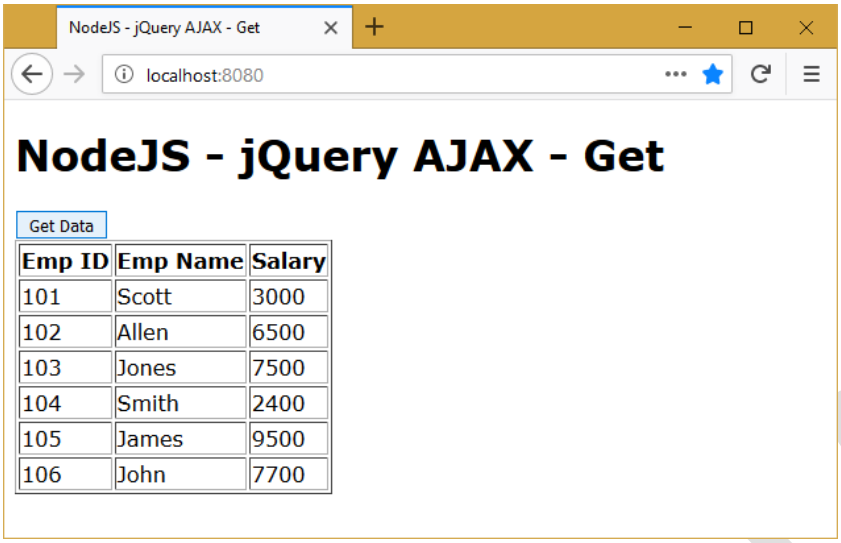
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Click on "GET data".



jQuery AJAX - NodeJS - Search - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "jquery": "latest",
      "express": "latest",
      "mongoose": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
}
app.use(express.static(__dirname));
```

```
//mongoose
var mongoose = require("mongoose");
var EmployeesSchema  = new mongoose.Schema({ empid: Number, empname: String, salary: Number });
var Employee = mongoose.model("employees", EmployeesSchema);

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//searchemployees
app.get("/searchemployees", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.query);
    mongoose.connect("mongodb://localhost/company");

    Employee.find({ "empname": { "$regex": request.query.str } }, fun3);
    function fun3(error, data)
    {
        if (error)
        {
            console.log(error);
            response.end();
        }
        else
        {
            console.log(data);
            response.header("content-type", "application/json");
            response.send(data);
            mongoose.connection.close();
        }
    }
}
```

c:\nodejs\index.html

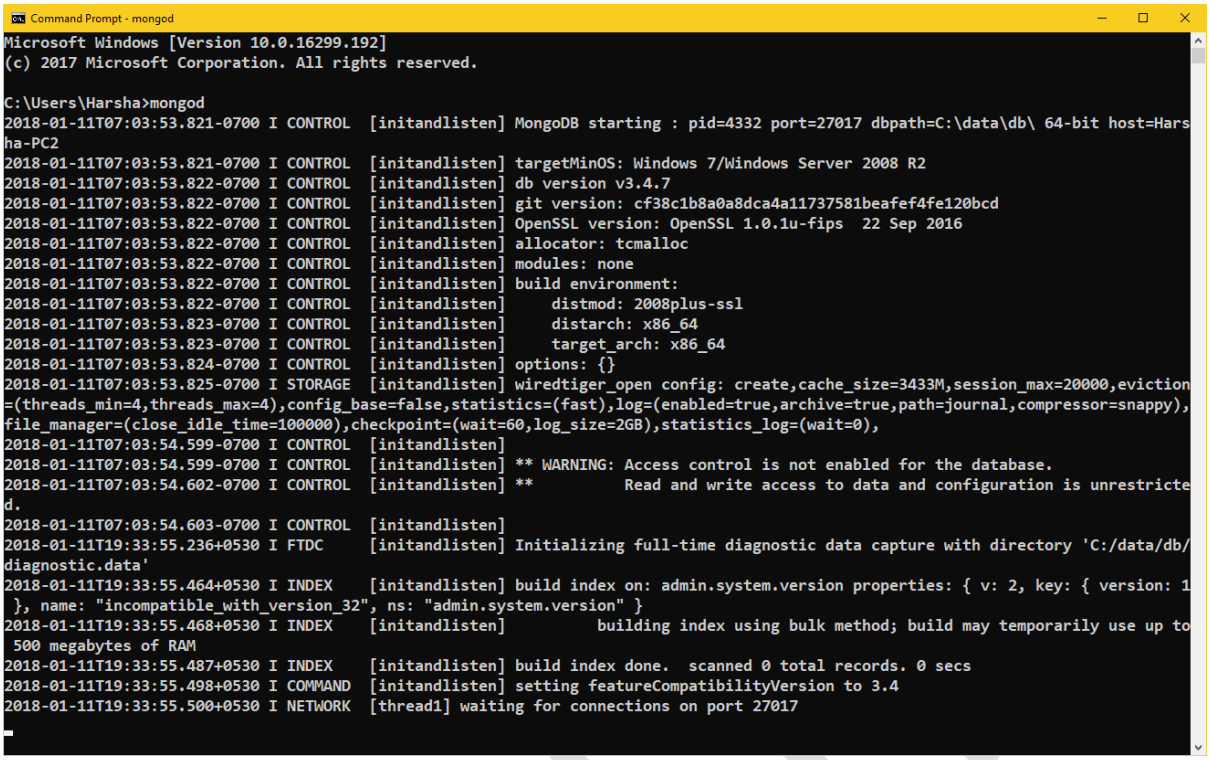
```
<!DOCTYPE html>
<html>
<head>
  <title>NodeJS - jQuery AJAX - Search</title>
</head>
<body>
  <h1>NodeJS - jQuery AJAX - Search</h1>
  <form>
    Search employees: <input type="text" id="txt1">
    <input type="submit" id="btn1" value="Search">
    <table id="table1" border="1">
      <tr> <th>Emp ID</th> <th>Emp Name</th> <th>Salary</th> </tr>
```

```
    </table>
  </form>

  <script src="node_modules/jquery/dist/jquery.js"></script>

  <script>
    $("#btn1").click(fun1);
    function fun1(event)
    {
      event.preventDefault();
      $.ajax({ type: "GET", url: "/searchemployees?str=" + $("#txt1").val(), success: fun2, error: fun3 });
    }

    function fun2(response)
    {
      $("#table1 tr:gt(0)").remove();
      for (var i = 0; i < response.length; i++)
      {
        $("#table1").append("<tr> <td>" + response[i].empid + "</td> <td>" + response[i].empname + "</td> <td>" +
response[i].salary + "</td> </tr>");
      }
    }

    function fun3(error)
    {
      alert(error);
    }
  </script>
</body>
</html>
```

Execution:

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```

```
Command Prompt - mongo
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>mongo
MongoDB shell version v3.4.7
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.7
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten]
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-01-11T07:03:54.602-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricte
d.
2018-01-11T07:03:54.603-0700 I CONTROL  [initandlisten]
> use company
switched to db company
> db.employees.insert([
... { "empid": 101, "empname": "Scott", "salary": 3000 },
... { "empid": 102, "empname": "Allen", "salary": 6500 },
... { "empid": 103, "empname": "Jones", "salary": 7500 },
... { "empid": 104, "empname": "Smith", "salary": 2400 },
... { "empid": 105, "empname": "James", "salary": 9500 },
... { "empid": 106, "empname": "John", "salary": 7700 }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

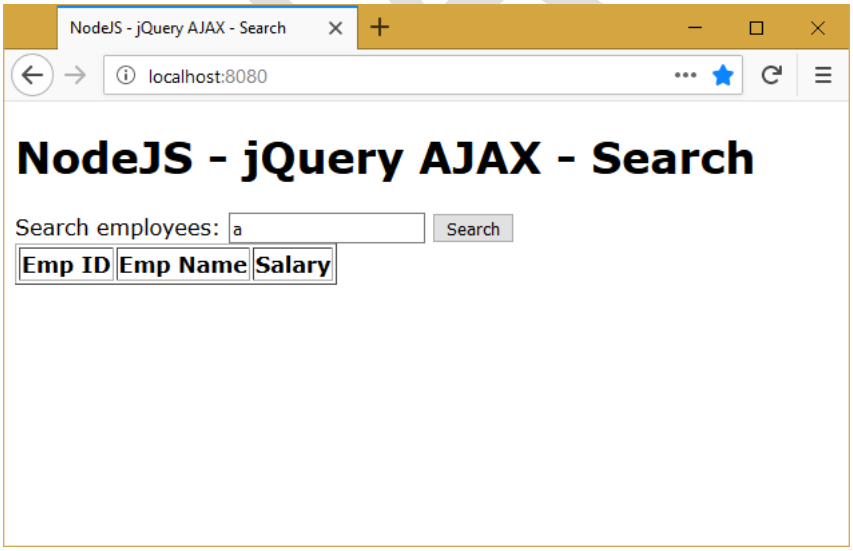
- Open Command Prompt 3

  cd c:\nodejs

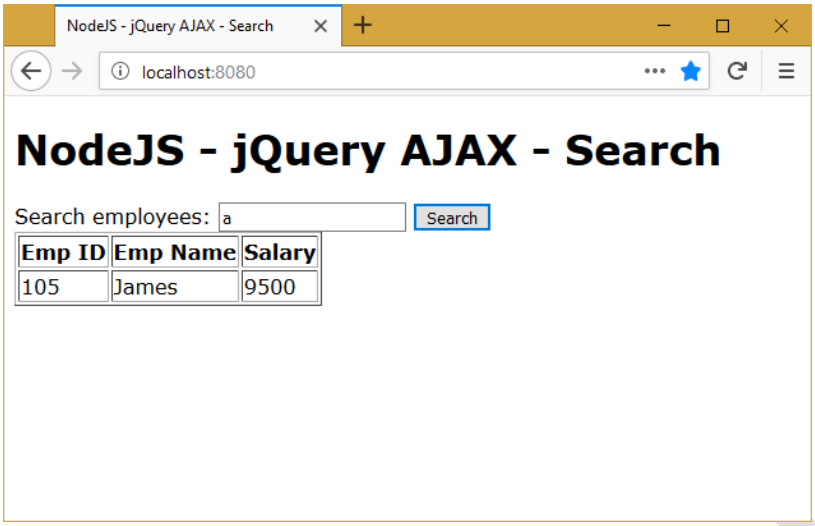
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Enter some text and click on "Search".



jQuery AJAX - NodeJS - Post - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "jquery": "latest",
      "body-parser": "latest",
      "express": "latest",
      "mongoose": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
```

```
    console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//mongoose
var mongoose = require("mongoose");
var EmployeeSchema  = new mongoose.Schema({ empid: Number, empname: String, salary: Number }, { versionKey:
false });
var Employee = mongoose.model("employees", EmployeeSchema);

//bodyparser
var bodyparser = require("body-parser");
app.use(bodyparser.urlencoded({ extended: true }));
app.use(bodyparser.json());

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//insertemployee
app.post("/insertemployee", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.body);
    mongoose.connect("mongodb://localhost/company");

    var newemp = new Employee({ empid: request.body.empid, empname: request.body.empname, salary:
request.body.salary});

    newemp.save(fun3);
    function fun3(error)
    {
        if (error)
        {
            console.log(error);
            response.end();
        }
        else
        {
            response.send("Successfully inserted");
            mongoose.connection.close();
        }
    }
}
```

c:\nodejs\index.html

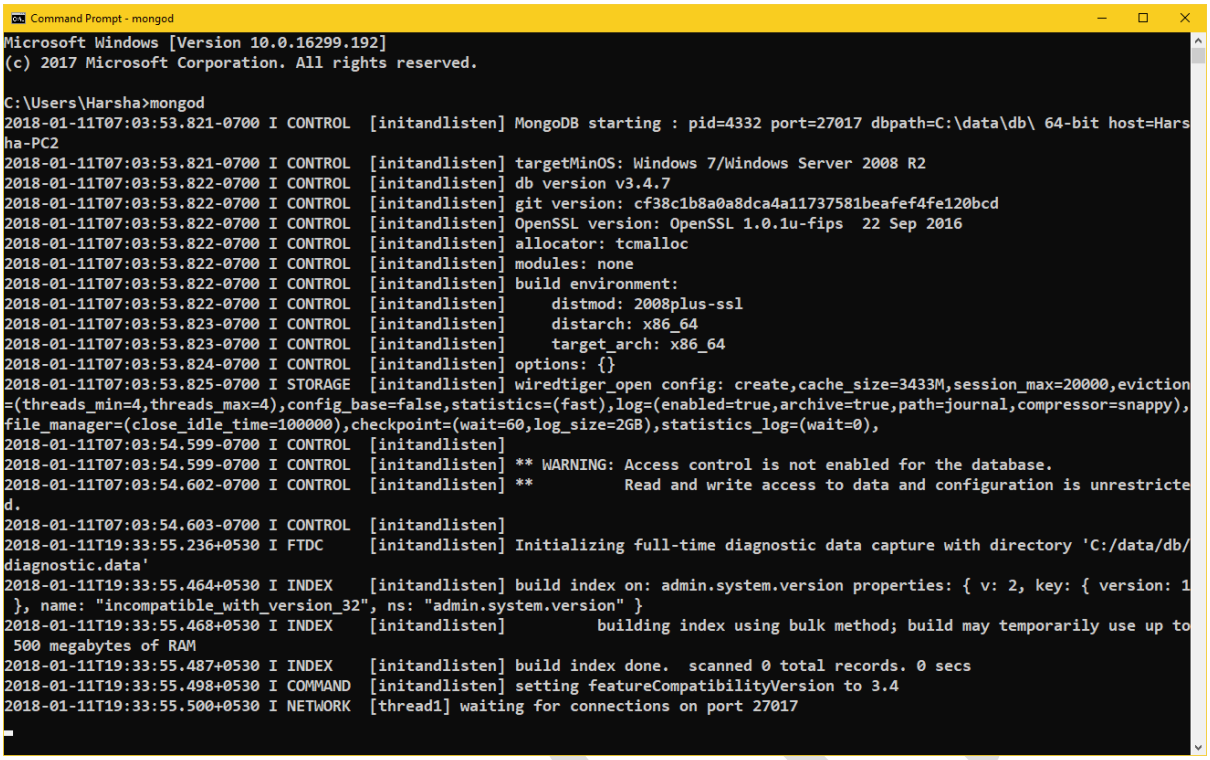
```
<!DOCTYPE html>
<html>
<head>
```

```
  <title>NodeJS - jQuery AJAX - Post</title>
</head>
<body>
  <h1>NodeJS - jQuery AJAX - Post</h1>
  <form>
    Emp ID: <input type="text" id="txt1"><br>
    Emp Name: <input type="text" id="txt2" /><br>
    Salary: <input type="text" id="txt3"><br>
    <input type="submit" id="btn1" value="Insert">
    <div id="div1"></div>
  </form>

  <script src="node_modules/jquery/dist/jquery.js"></script>

  <script>
    $("#btn1").click(fun1);
    function fun1(event)
    {
      event.preventDefault();
      var mydata = { "empid": $("#txt1").val(), "empname": $("#txt2").val(), "salary": $("#txt3").val() };
      $.ajax({ type: "POST", url: "/insertemployee", success: fun2, error: fun3, data: mydata });
    }

    function fun2(response)
    {
      $("#div1").html(response);
    }

    function fun3(error)
    {
      alert(error);
    }
  </script>
</body>
</html>
```

Execution:

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```

```
Command Prompt - mongo
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>mongo
MongoDB shell version v3.4.7
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.7
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten]
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-01-11T07:03:54.602-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricte
d.
2018-01-11T07:03:54.603-0700 I CONTROL  [initandlisten]
> use company
switched to db company
> db.employees.insert([
... { "empid": 101, "empname": "Scott", "salary": 3000 },
... { "empid": 102, "empname": "Allen", "salary": 6500 },
... { "empid": 103, "empname": "Jones", "salary": 7500 },
... { "empid": 104, "empname": "Smith", "salary": 2400 },
... { "empid": 105, "empname": "James", "salary": 9500 },
... { "empid": 106, "empname": "John", "salary": 7700 }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

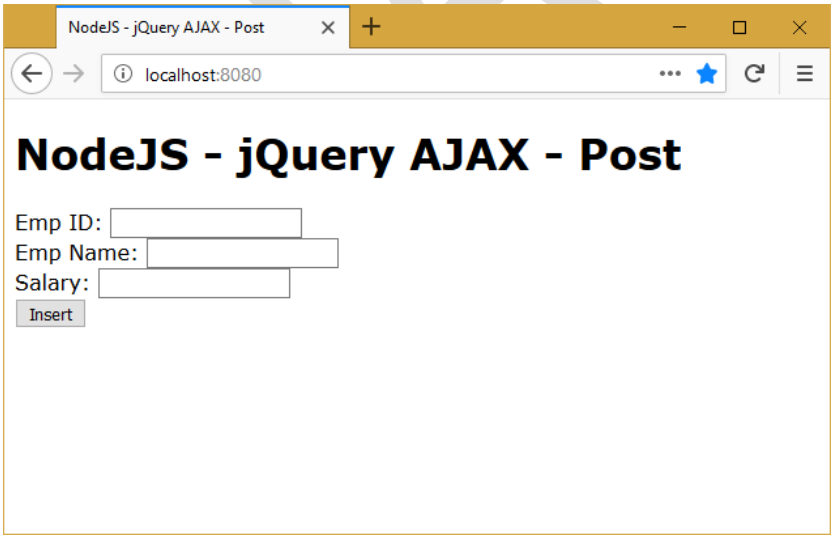
- Open Command Prompt 3

  cd c:\nodejs

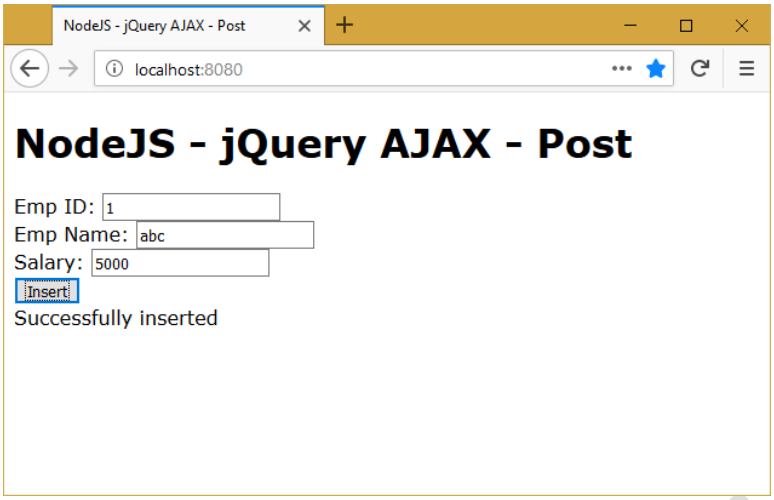
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Enter some details and click on "Insert".



jQuery AJAX - NodeJS - Put - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "jquery": "latest",
      "body-parser": "latest",
      "express": "latest",
      "mongoose": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
}
app.use(express.static(__dirname));
```

```
//mongoose
var mongoose = require("mongoose");
var EmployeeSchema  = new mongoose.Schema({ "empid": Number, "empname": String, "salary": Number }, {
versionKey: false });
var Employee = mongoose.model("employees", EmployeeSchema);

//bodyparser
var bodyparser = require("body-parser");
app.use(bodyparser.urlencoded({ extended: true }));
app.use(bodyparser.json());

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//updateemployee
app.put("/updateemployee", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.body);
    mongoose.connect("mongodb://localhost/company");

    Employee.findOne({empid: request.body.empid}, fun3);
    function fun3(err, data)
    {
        if(err)
        {
            response.send("Error");
        }
        else
        {
            data.empname = request.body.empname;
            data.salary = request.body.salary;

            data.save(fun4);
            function fun4(error)
            {
                if(error)
                {
                    response.send("Not updated");
                }
                else
                {
                    response.send("Successfully Updated");
                }
                mongoose.connection.close();
            }
        }
    }
```

```
}
```

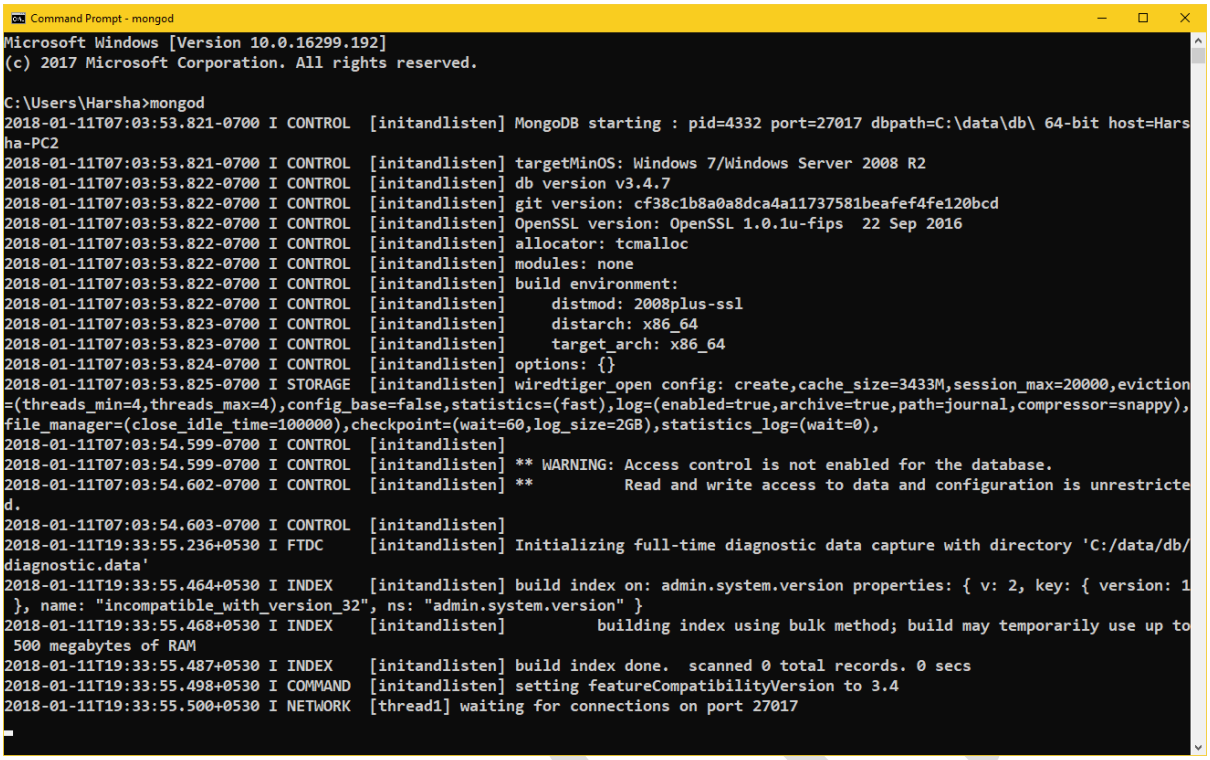
c:\nodejs\index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>NodeJS - jQuery AJAX - Put</title>
</head>
<body>
  <h1>NodeJS - jQuery AJAX - Put</h1>
  <form>
    Existing Emp ID: <input type="text" id="txt1"><br>
    Emp Name: <input type="text" id="txt2"><br>
    Salary: <input type="text" id="txt3"><br>
    <input type="submit" id="btn1" value="Update">
    <div id="div1"></div>
  </form>

  <script src="node_modules/jquery/dist/jquery.js"></script>

  <script>
    $("#btn1").click(fun1);
    function fun1(event)
    {
      event.preventDefault();
      var mydata = { "empid": $("#txt1").val(), "empname": $("#txt2").val(), "salary": $("#txt3").val() };
      $.ajax({ type: "PUT", url: "/updateemployee", data: mydata, success: fun2, error: fun3 });
    }

    function fun2(response)
    {
      $("#div1").html(response);
    }

    function fun3(error)
    {
      alert(error);
    }
  </script>
</body>
</html>
```

Execution:

- Open Command Prompt 1

  mongod



- Open Command Prompt 2

```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```

```
Command Prompt - mongo
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>mongo
MongoDB shell version v3.4.7
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.7
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten]
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-01-11T07:03:54.602-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricte
d.
2018-01-11T07:03:54.603-0700 I CONTROL  [initandlisten]
> use company
switched to db company
> db.employees.insert([
... { "empid": 101, "empname": "Scott", "salary": 3000 },
... { "empid": 102, "empname": "Allen", "salary": 6500 },
... { "empid": 103, "empname": "Jones", "salary": 7500 },
... { "empid": 104, "empname": "Smith", "salary": 2400 },
... { "empid": 105, "empname": "James", "salary": 9500 },
... { "empid": 106, "empname": "John", "salary": 7700 }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

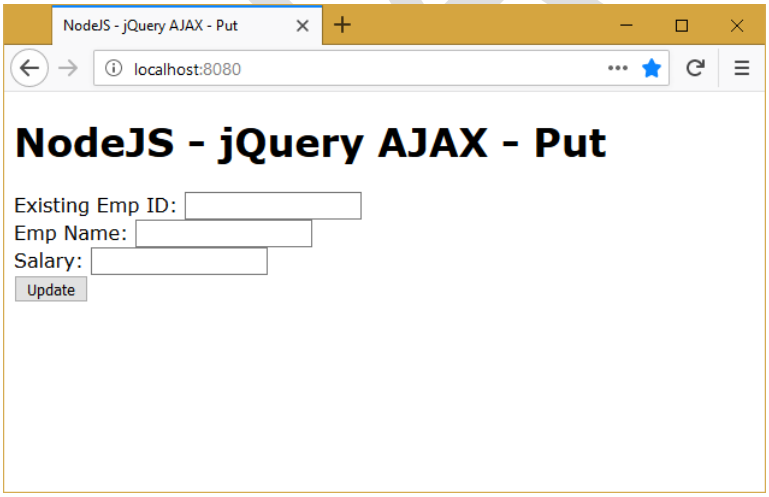
- Open Command Prompt 3

  cd c:\nodejs

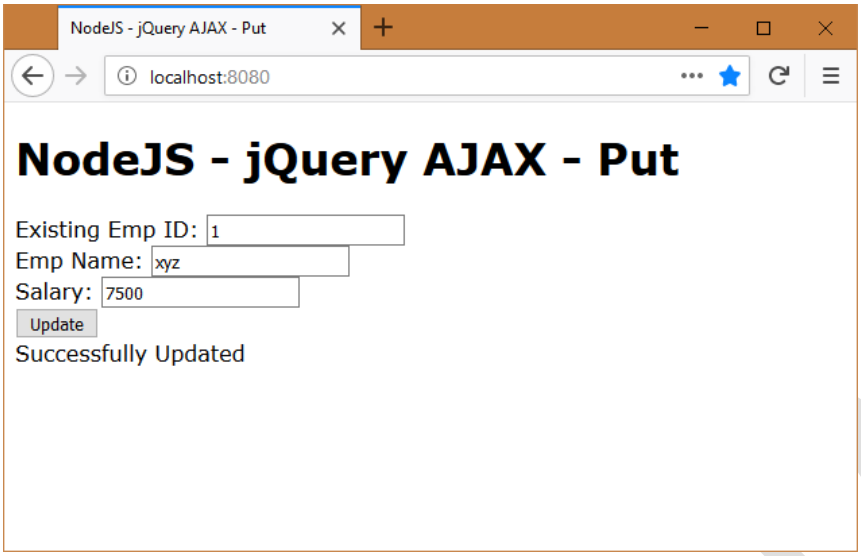
  npm install

  node httpserver.js

- Open browser and enter the url: http://localhost:8080



Enter some details and click on "Update".



jQuery AJAX - NodeJS - Delete - Example

Creating the application

- Create "c:\nodejs" folder.
- Create "httpserver.js" and "index.html" files in the "c:\nodejs" folder.

c:\nodejs

- httpserver.js
- index.html
- package.json

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "jquery": "latest",
      "body-parser": "latest",
      "express": "latest",
      "mongoose": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
```

```
}
app.use(express.static(__dirname));

//mongoose
var mongoose = require("mongoose");
var EmployeeSchema = new mongoose.Schema({ empid: Number, empname: String, salary: Number }, { versionKey:
false });
var Employee = mongoose.model("employees", EmployeeSchema);

//bodyparser
var bodyparser = require("body-parser");
app.use(bodyparser.urlencoded({ extended: true }));
app.use(bodyparser.json());

//home
app.get("/", fun1);
function fun1(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    response.sendFile(__dirname + "/index.html");
}

//deleteemployee
app.delete("/deleteemployee/:empid", fun2);
function fun2(request, response)
{
    console.log("request received at " + request.url + ", " + request.method);
    console.log(request.params);
    mongoose.connect("mongodb://localhost/company");

    Employee.remove({empid: request.params.empid}, fun3);
    function fun3(err)
    {
        if(err)
        {
            response.send("Not deleted");
        }
        else
        {
            response.send("Successfully Deleted");
        }
        mongoose.connection.close();
    }
}
```

c:\nodejs\index.html

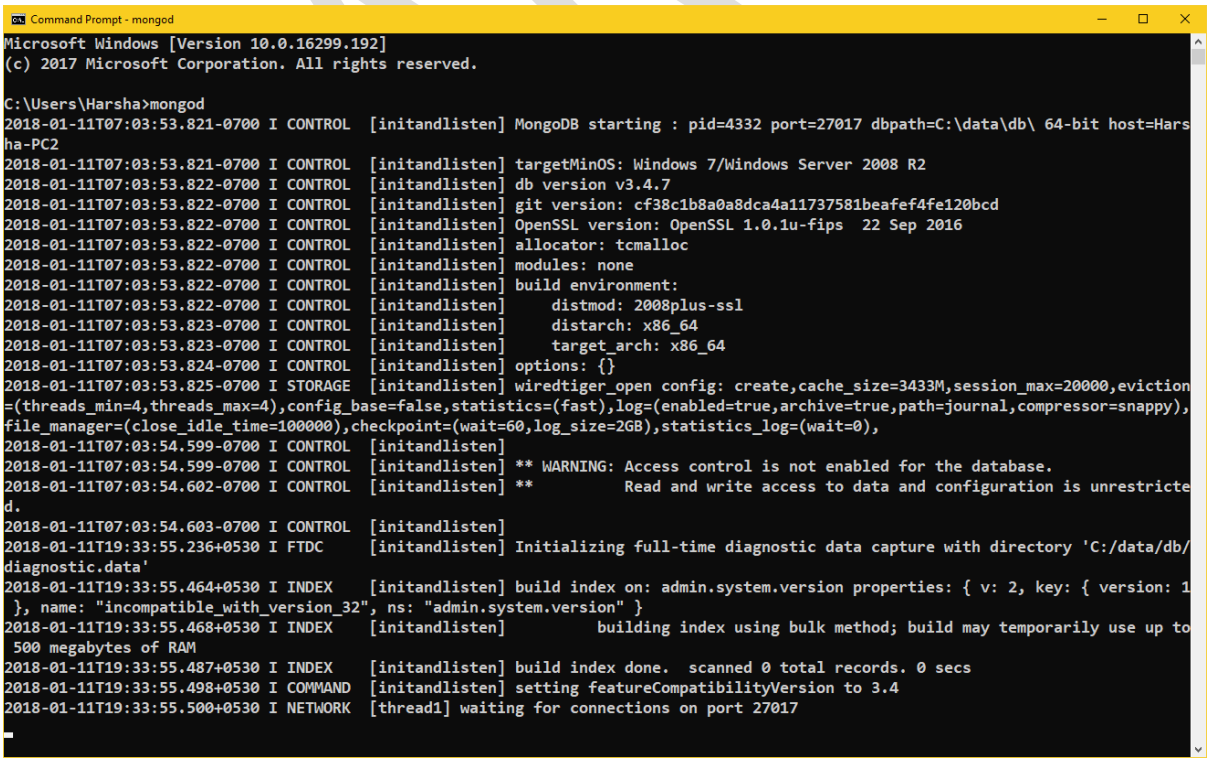
```
<!DOCTYPE html>
<html>
<head>
  <title>NodeJS - jQuery AJAX -Delete</title>
</head>
<body>
  <h1>NodeJS - jQuery AJAX -Delete</h1>
  <form>
    Existing Emp ID: <input type="text" id="txt1"><br>
```

```
    <input type="submit" id="btn1" value="Delete">
    <div id="div1"></div>
  </form>

  <script src="node_modules/jquery/dist/jquery.js"></script>

  <script>
    $("#btn1").click(fun1);
    function fun1(event)
    {
      event.preventDefault();
      $.ajax({ type: "DELETE", url: "/deleteemployee/" + $("#txt1").val(), success: fun2, error: fun3 });
    }

    function fun2(response)
    {
      $("#div1").html(response);
    }

    function fun3(error)
    {
      alert(error);
    }
  </script>
</body>
</html>
```

Execution:

- Open Command Prompt 1

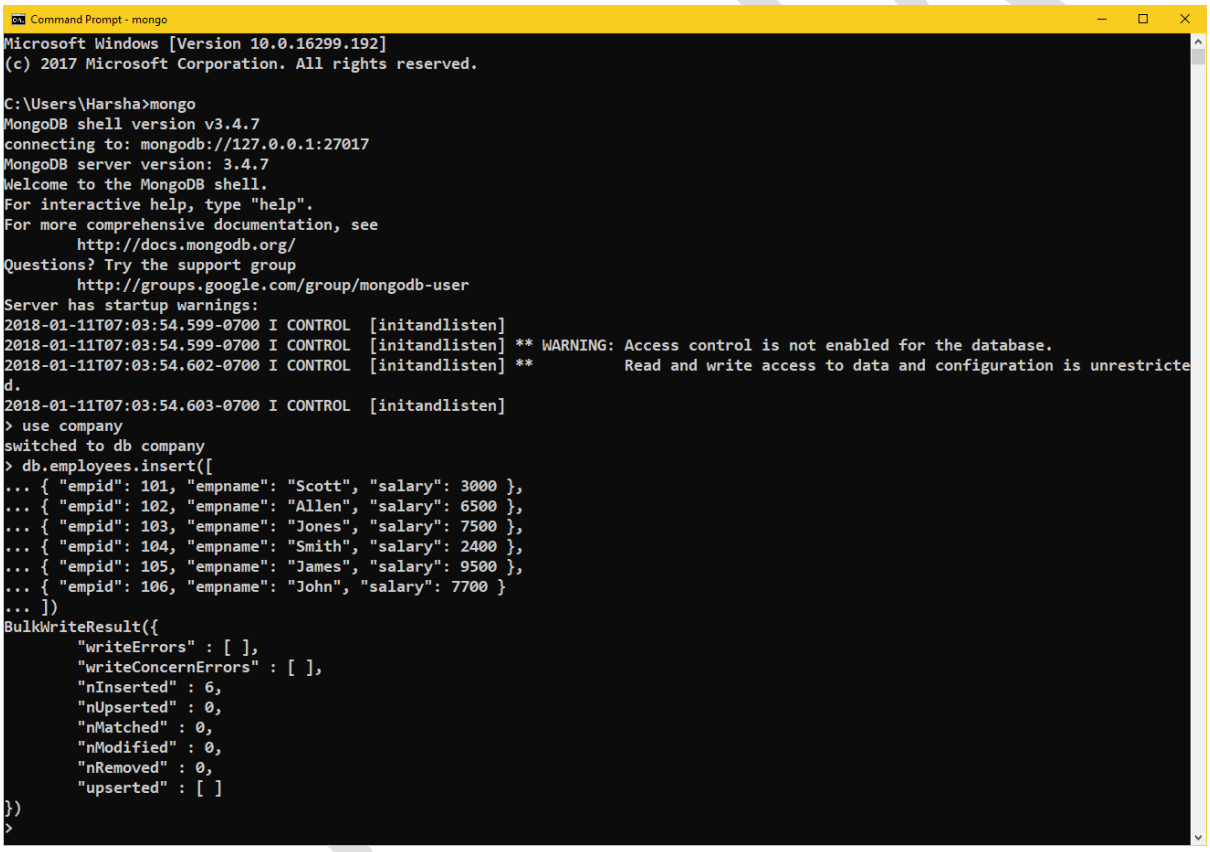
  mongod



```
Command Prompt - mongod
Microsoft Windows [Version 10.0.16299.192]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Harsha>mongod
2018-01-11T07:03:53.821-0700 I CONTROL  [initandlisten] MongoDB starting : pid=4332 port=27017 dbpath=C:\data\db\ 64-bit host=Hars
ha-PC2
2018-01-11T07:03:53.821-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-01-11T07:03:53.822-0700 I CONTROL  [initandlisten] db version v3.4.7
2018-01-11T07:03:53.822-0700 I CONTROL  [initandlisten] git version: cf38c1b8a0a8dca4a11737581beafef4fe120bcd
2018-01-11T07:03:53.822-0700 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0.1u-fips  22 Sep 2016
2018-01-11T07:03:53.822-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2018-01-11T07:03:53.822-0700 I CONTROL  [initandlisten] modules: none
2018-01-11T07:03:53.822-0700 I CONTROL  [initandlisten] build environment:
2018-01-11T07:03:53.822-0700 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2018-01-11T07:03:53.823-0700 I CONTROL  [initandlisten]     distarch: x86_64
2018-01-11T07:03:53.823-0700 I CONTROL  [initandlisten]     target_arch: x86_64
2018-01-11T07:03:53.824-0700 I CONTROL  [initandlisten] options: {}
2018-01-11T07:03:53.825-0700 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=3433M,session_max=20000,eviction
=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),
file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten]
2018-01-11T07:03:54.599-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-01-11T07:03:54.602-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricte
d.
2018-01-11T07:03:54.603-0700 I CONTROL  [initandlisten]
2018-01-11T19:33:55.236+0530 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C:/data/db/
diagnostic.data'
2018-01-11T19:33:55.464+0530 I INDEX    [initandlisten] build index on: admin.system.version properties: { v: 2, key: { version: 1
 }, name: "incompatible_with_version_32", ns: "admin.system.version" }
2018-01-11T19:33:55.468+0530 I INDEX    [initandlisten]          building index using bulk method; build may temporarily use up to
 500 megabytes of RAM
2018-01-11T19:33:55.487+0530 I INDEX    [initandlisten] build index done.  scanned 0 total records. 0 secs
2018-01-11T19:33:55.498+0530 I COMMAND  [initandlisten] setting featureCompatibilityVersion to 3.4
2018-01-11T19:33:55.500+0530 I NETWORK  [thread1] waiting for connections on port 27017
```

- Open Command Prompt 2

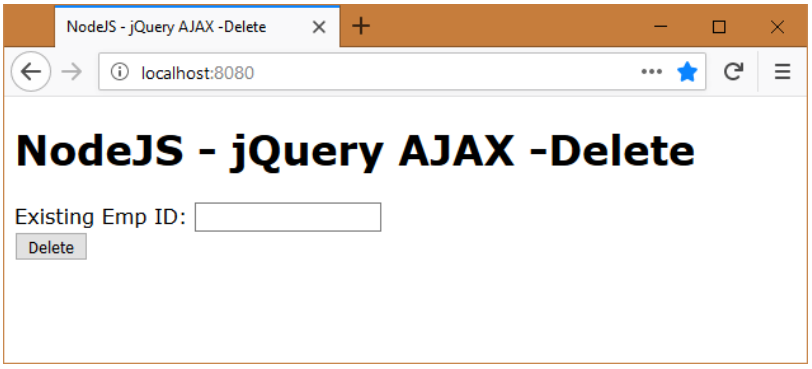
```
mongo
use company
db.employees.insert([
{ "empid": 101, "empname": "Scott", "salary": 3000 },
{ "empid": 102, "empname": "Allen", "salary": 6500 },
{ "empid": 103, "empname": "Jones", "salary": 7500 },
{ "empid": 104, "empname": "Smith", "salary": 2400 },
{ "empid": 105, "empname": "James", "salary": 9500 },
{ "empid": 106, "empname": "John", "salary": 7700 }
])
```



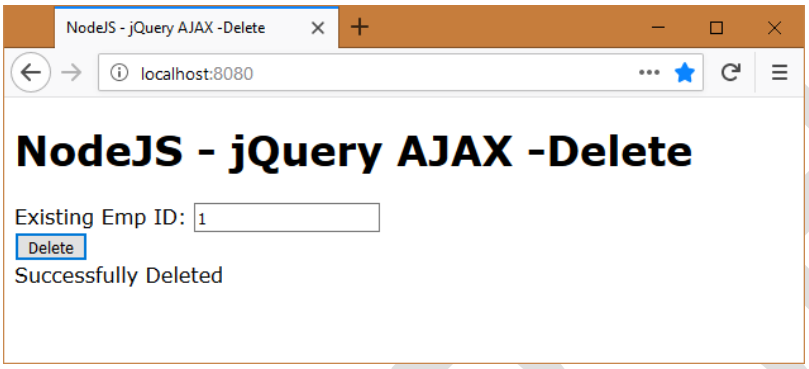
- Open Command Prompt 3

```
cd c:\nodejs
npm install
node httpserver.js
```

- Open browser and enter the url: http://localhost:8080



Enter some emp id and click on "Delete".



MVC

Introduction to MVC

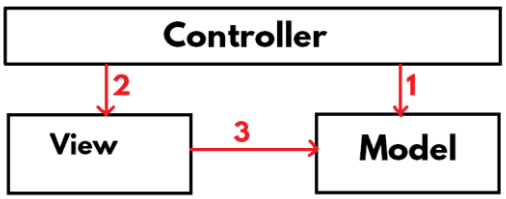
- "MVC" is a design pattern / architecture.
    1. **Design pattern:** Solution for known problem
    2. **Architecture:** Pictural representation of components.
- "MVC" was introduced in 1979, by "Reenskaug" in "SmallTalk" language.
- To create applications using "MVC architecture", we require "MVC frameworks".
    1. Java : Spring
    2. PHP : Cake PHP
    3. NodeJS : Express
    4. JavaScript : AngularJS 1
    5. iOS : Objective C
    6. .NET : ASP.NET MVC

- **Advantages of MVC:**
    1. Clean separation between data, functionality logic and presentation logic.

MVC Architecture

- MVC architecture contains three components:
    1. Model : The JS Object that stores data.

2. View : The web page that displays model data.
3. Controller : The JS function that loads data into model.



1. Controller calls Model.
2. Controller calls View.
3. View calls Model.

View Engines

- View Engine is a package that specifies set of syntaxes to create views.
- It contains its own view file extension.
- It specifies syntax to print model data in the view.
- It executes the view (replaces expressions with actual model data) and returns the view result; so it can be sent as response to the browser by the controller.
- **List of View Engines:**
    1. EJS
    2. Mustache
    3. Handlebars
    4. PUG

EJS

- EJS stands for "Embedded JavaScript".
- The "EJS" package is a view engine, based on which you can create views.
- View is a web page that receives a javascript object at run time and displays the same data in the appropriate places in the web page.
- File extension of EJS view: ".ejs".

Steps for working with EJS

- **Import the "ejs" package in "package.json" file:**

```
"dependencies":
{
  "ejs": "latest"
}
```

- **Set the view engine as "ejs":**

```
var ejs = require("ejs");
app.set("view engine", "ejs");
```

- **Create model:**

```
var data = { property: value, property: value, ... };
```

- **Call the view and pass model to the view:**

```
response.render("filename.ejs", data);
```

- **EJS - Code Blocks:**

```
<%
  javascript code here (server side)
%>
```

- **EJS - Expressions:**

```
<%=property%>
```

- **EJS - If:**

```
<% if (condition) { %>
    html code here
<% } else { %>
    html code here
<% } %>
```

- **EJS - For Loop:**

```
<%for (initialization; condition; iteration) { %>
    html code here
<% } %>
```

- **EJS - Calling Partial View:**

```
<%include filename.ejs%>
```

EJS - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "express": "latest",
      "ejs": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
  console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//ejs
var ejs = require("ejs");
app.set("view engine", "ejs");

//home
app.get("/", fun1);
```

```
function fun1(request, response)
{
  var data =
  {
    firstname: "adam",
    lastname: "smith",
    marks: 80,
    address: { city: "Hyderabad", country: "India" },
    pass: true,
    subjects:
    [
      { subjectname: "Maths", grade: "O" },
      { subjectname: "Physics", grade: "A" },
      { subjectname: "Chemistry", grade: "B" }
    ]
  };
  response.render("student.ejs", data);
}
```

c:\nodejs\views\student.ejs

```
<html>
 <head>
  <title>EJS</title>
 </head>
 <body>
  <h1>EJS</h1>
  <p>
   Firstname: <%=firstname%><br>
   Lastname: <%=lastname%><br>
   Marks: <%=marks%><br>
   City: <%=address.city%><br>
   Country: <%=address.country%><br>
  </p>

  <%if(pass == true) {%>
   <p>Congratulations</p>
  <%} else {%>
   <p>Better luck next time</p>
  <%}%>

  <table border="1">
   <tr>
    <th>Subject Name</th>
    <th>Grade</th>
   </tr>
   <%for(var i = 0; i < subjects.length; i++) {%>
    <tr>
     <td><%=subjects[i].subjectname%></td>
     <td><%=subjects[i].grade%></td>
    </tr>
   <%}%>
  </table>

  <%include sample.ejs%>
 </body>
```

```
</html>
```

c:\nodejs\views\sample.ejs

```
<h1>heading</h1>
<p>paragraph</p>
```

Execution

- Open Command Prompt

  cd c:\nodejs

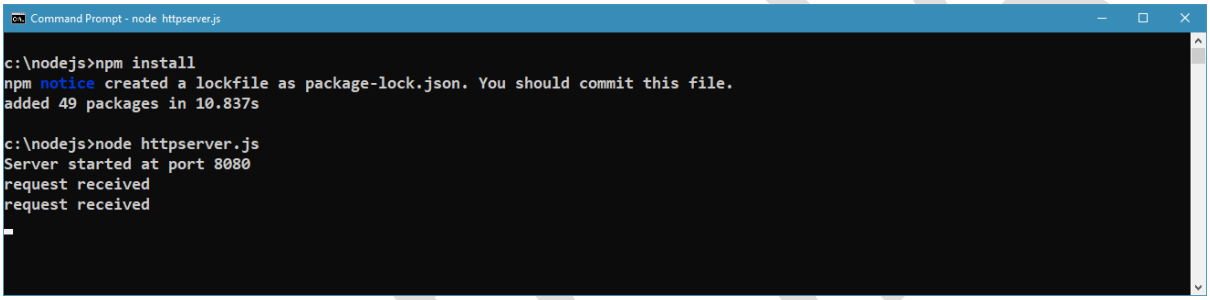
  npm install

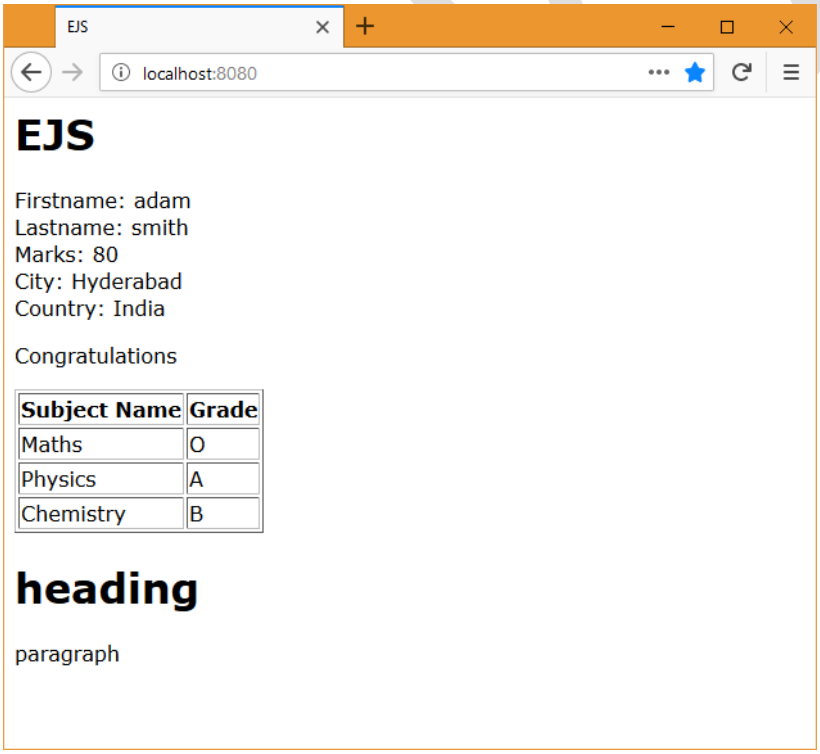
  node httpserver.js

- Open any browser

  http://localhost:8080

Output:





Mustache

- The "mustache" package is a view engine, based on which you can create the view.

- File extension of mustache view: ".mustache".

Steps for working with Mustache

- **Import the "mustache" package in "package.json" file:**

```
"dependencies":
{
  "mustache-express": "latest"
}
```

- **Set the view engine as "mustache":**

```
var mustacheexpress = require("mustache-express");
app.engine("mustache", mustacheexpress());
app.set("view engine", "mustache");
```

- **Create model:**

```
var data = { property: value, property: value, ... };
```

- **Call the view and pass model to the view:**

```
response.render("filename.mustache", data);
```

- **Mustache – Expressions:**

```
{{property}}
```

- **Mustache – If:**

```
{{#property}}
    html code here
{{/property}}
{{^property}}
    html code here
{{/property}}
```

- **Mustache – For Loop:**

```
{{#property}}
  html code here
{{/property}}
```

- **Mustache – Calling Partial View:**

```
{{>filename.mustache}}
```

Mustache - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "express": "latest",
      "mustache-express": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
```

```
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
  console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//mustache-express
var mustacheexpress = require("mustache-express");
app.engine("mustache", mustacheexpress());
app.set("view engine", "mustache");

//home
app.get("/", fun1);
function fun1(request, response)
{
  var data =
  {
    firstname: "adam",
    lastname: "smith",
    marks: 80,
    address: { city: "Hyderabad", country: "India" },
    pass: true,
    subjects:
    [
      { subjectname: "Maths", grade: "O" },
      { subjectname: "Physics", grade: "A" },
      { subjectname: "Chemistry", grade: "B" }
    ]
  };
  response.render("student.mustache", data);
}
```

c:\nodejs\views\student.mustache

```
<html>
  <head>
    <title>Mustache</title>
  </head>
  <body>
    <h1>Mustache</h1>
    <p>
      Firstname: {{firstname}}<br>
      Lastname: {{lastname}}<br>
      Marks: {{marks}}<br>
      City: {{address.city}}<br>
      Country: {{address.country}}<br>
    </p>

    {{#pass}}
    <p>Congratulations</p>
    {{/pass}}
    {{^pass}}
    <p>Better luck next time</p>
```

```
    {{/pass}}

    <table border="1">
      <tr>
        <th>Subject Name</th>
        <th>Grade</th>
      </tr>
      {{#subjects}}
        <tr>
          <td>{{subjectname}}</td>
          <td>{{grade}}</td>
        </tr>
      {{/subjects}}
    </table>

    {{>sample}}
  </body>
</html>
```

c:\nodejs\views\sample.mustache

```
<h1>heading</h1>
<p>paragraph</p>
```

Execution

- Open Command Prompt

  cd c:\nodejs

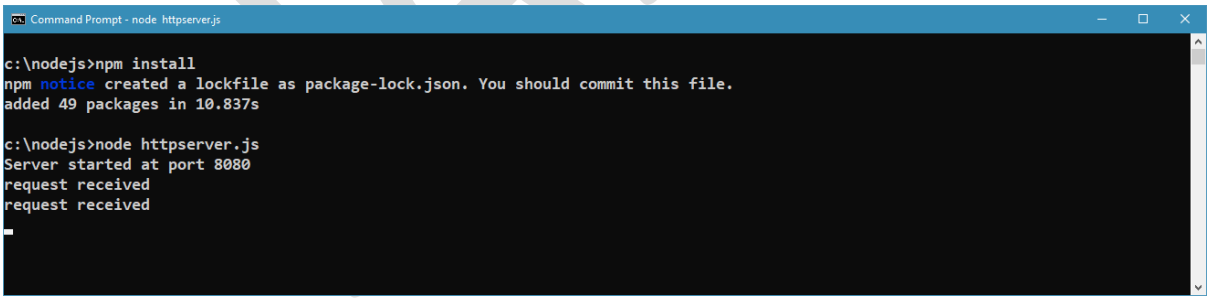
  npm install

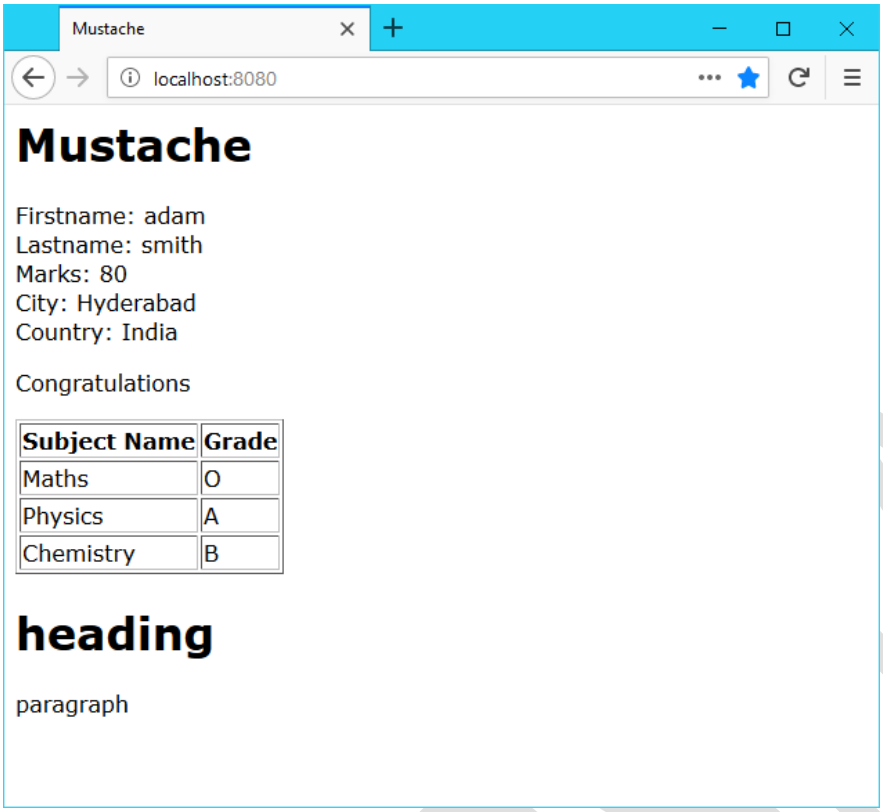
  node httpserver.js

- Open any browser

  http://localhost:8080

Output:





Handlebars

- The "handlebars" package is a view engine, based on which you can create the view.
- File extension of handlebars view: ".hbs".

Steps for working with Handlebars

- **Import the "handlebars" package in "package.json" file:**

```
"dependencies":
{
  "express-handlebars": "latest"
}
```

- **Set the view engine as "handlebars":**

```
var expresshandlebars = require("express-handlebars");
app.engine("hbs", expresshandlebars());
app.set("view engine", "hbs");
```

- **Create model:**

```
var data = { property: value, property: value, ... };
```

- **Call the view and pass model to the view:**

```
response.render("filename.hbs", data);
```

- **Handlebars - Expressions:**

```
{{property}}
```

- **Handlebars - With:**

```
{{#with property}}
    html code here
```

```
{{/with}}
```

- **Handlebars - If:**

```
{{#if property}}
    html code here
{{else}}
    html code here
{{/if}}
```

- **Handlebars - For Loop:**

```
{{#each property}}
  html code here
{{/each}}
```

- **Handlebars - Calling Partial View:**

```
{{>filename}}
```

Handlebars - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "express": "latest",
      "express-handlebars": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
  console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//express-handlebars
var expresshandlebars = require("express-handlebars");
app.engine("hbs", expresshandlebars());
app.set("view engine", "hbs");

//home
app.get("/", fun1);
function fun1(request, response)
{
  var data =
  {
    firstname: "adam",
    lastname: "smith",
```

```
    marks: 80,
    address: { city: "Hyderabad", country: "India" },
    pass: true,
    subjects:
    [
      { subjectname: "Maths", grade: "O" },
      { subjectname: "Physics", grade: "A" },
      { subjectname: "Chemistry", grade: "B" }
    ]
  };
  response.render("student.hbs", data);
}
```

c:\nodejs\views\student.hbs

```
<html>
  <head>
    <title>Handlebars</title>
  </head>
  <body>
    <h1>Handlebars</h1>
    <p>
      Firstname: {{firstname}}<br>
      Lastname: {{lastname}}<br>
      Marks: {{marks}}<br>
      {{#with address}}
        City: {{city}}<br>
        Country: {{country}}<br>
      {{/with}}
    </p>

    {{#if pass}}
    <p>Congratulations</p>
    {{else}}
    <p>Better luck next time</p>
    {{/if}}

    <table border="1">
      <tr>
        <th>Subject Name</th>
        <th>Grade</th>
      </tr>
          {{#each subjects}}
            <tr>
              <td>{{subjectname}}</td>
              <td>{{grade}}</td>
            </tr>
          {{/each}}
        </table>

        {{>sample}}
      </body>
    </html>
```

c:\nodejs\views\partials\sample.handlebars

```
<h1>heading</h1>
```

```
<p>paragraph</p>
```

Execution

- Open Command Prompt

  cd c:\nodejs

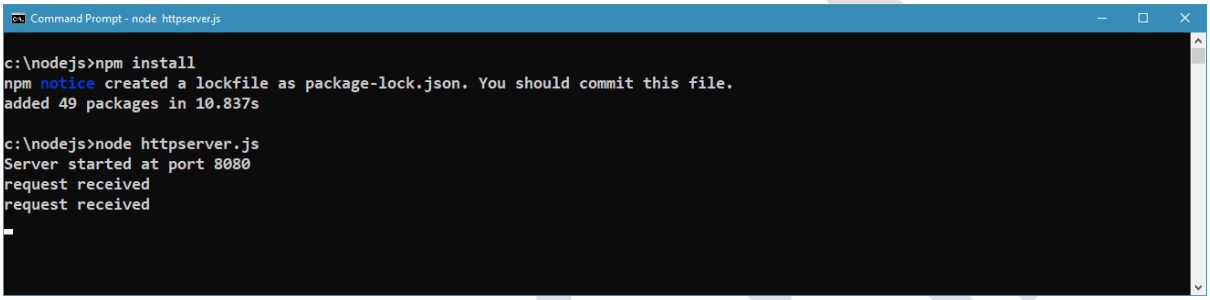
  npm install

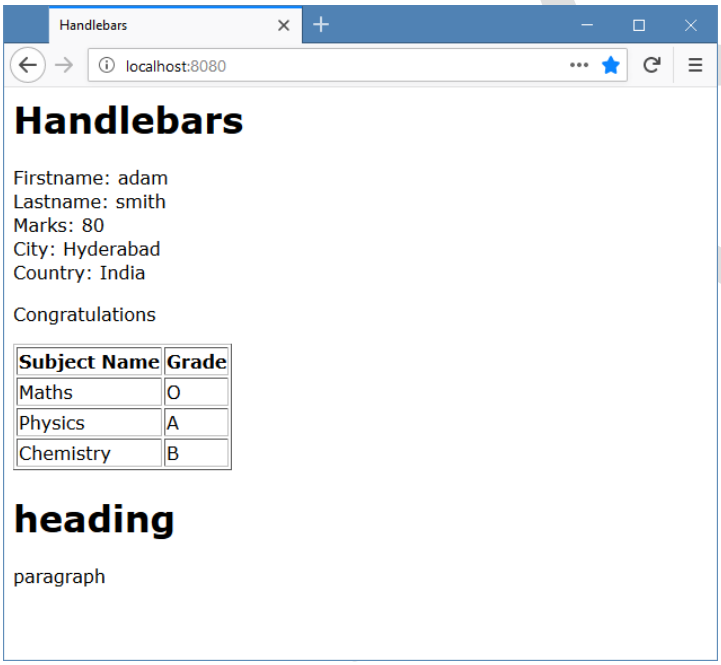
  node httpserver.js

- Open any browser

  http://localhost:8080

Output:





PUG

- The "pug" package is a view engine, based on which you can create the view.
- "Pug" was known as "jade" earlier.
- File extension of pug view: ".pug".

Steps for working with Pug

- **Import the "pug" package in "package.json" file:**

  "dependencies":

```
{
    "pug": "latest"
}
```

- **Set the view engine as "pug":**

```
var pug = require("pug");
app.set("view engine", "pug");
```

- **Create model:**
  var data = { property: value, property: value, … };

- **Call the view and pass model to the view:**
  response.render("filename.pug", data);

- **Pug – DOCTYPE:**
  doctype html

- **Pug – Tag:**
  tag

- **Pug – Tag with Single-line content:**
  tag content

- **Pug – Tag with Multi-line content:**

```
tag.
  line 1
  line 2
```

- **Pug – Tag with Dynamic Content:**
  tag=property

- **Pug – Expressions:**
  #{property}

- **Pug – If:**

```
if property
    content
else
    content
```

- **Pug – Tag with ID:**
  tag#id

- **Pug – Tag with Class:**
  tag.class

- **Pug – Tag with attributes:**
  tag(attribute="value")

- **Pug – For Loop:**

```
for variable in property
    content
```

- **Pug – Calling Partial View:**
  include filename.pug

Pug - Example

c:\nodejs\package.json

```
{
   "name": "mypackage",
   "version": "1.0.0",
   "description": "this is my package",
   "license": "ISC",
   "repository": "none",
   "dependencies":
   {
       "express": "latest",
       "pug": "latest"
   }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
  console.log("Server started at port 8080");
}
app.use(express.static(__dirname));

//pug
var pug = require("pug");
app.set("view engine", "pug");

//home
app.get("/", fun1);
function fun1(request, response)
{
  var data =
  {
    firstname: "adam",
    lastname: "smith",
    marks: 80,
    address: { city: "Hyderabad", country: "India" },
    pass: true,
    subjects:
    [
      { subjectname: "Maths", grade: "O" },
      { subjectname: "Physics", grade: "A" },
      { subjectname: "Chemistry", grade: "B" }
    ]
  };
  response.render("student.pug", data);
}
```

c:\nodejs\views\student.pug

```
html
 head
  title Pug
```

```
body
 h1 Pug
 p
  span Firstname: #{firstname}
  br
  span Lastname: #{lastname}
  br
  span Marks: #{marks}
  br
  span City: #{address.city}
  br
  span Country: #{address.country}
  br

 p
  if pass
   p Congratulations
  else
   p Better luck next time

 table(border="1")
  tr
   th Subject Name
   th Grade
  for subject in subjects
   tr
    td #{subject.subjectname}
    td #{subject.grade}

 include sample.pug
```

c:\nodejs\views\sample.pug

```
h1 heading
p paragraph
```

Execution

- Open Command Prompt

  cd c:\nodejs

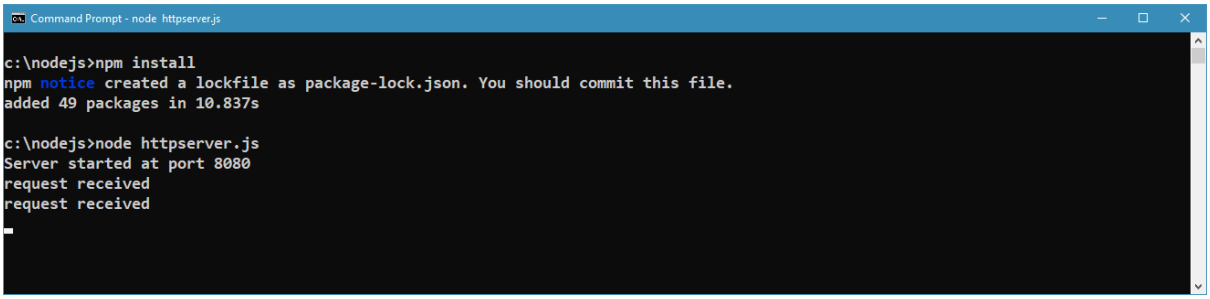
  npm install

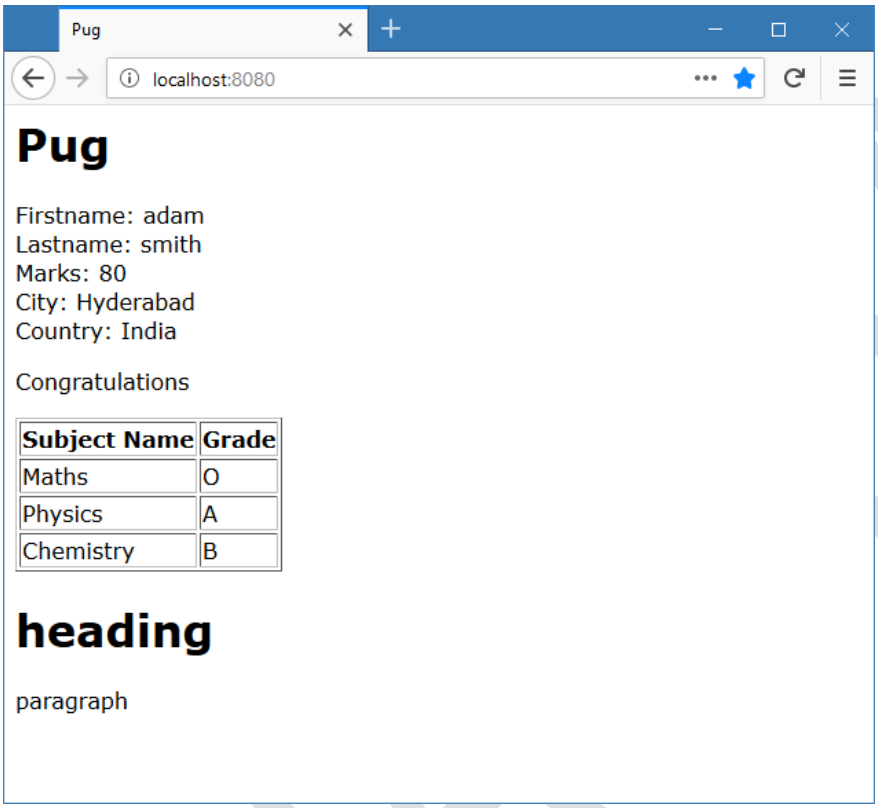
  node httpserver.js

- Open any browser

  http://localhost:8080

Output:





Cookies and Session

Cooke-parser Package

- The "cookie-parser" package is used to convert the request cookies into "javascript object" format.
- Cookie is a temporary file that will be stored in the browser memory, which will be submitted to the server automatically for each request.
- The "nodejs program" (server side program) can create, retrieve and delete the cookies, by using "cookie-parser" package.

Steps for working with "cookie-parser"

- **Import the "cookie-parser" package in "package.json" file:**

```
"dependencies":
{
  "cookie-parser": "latest"
}
```

- **Acquire "cookie-parser" module:**

```
var cookieparser = require("cookie-parser");
```

- **Enable cookie parser:**
  app.use(cookieparser());

- **Create a cookie and send it to the browser:**
  response.cookie("key", "value", { options });
  Options:
  maxAge : Represents the lifetime of the cookie, in the form of milli seconds.
  httpOnly : true / false. If "true" it disables accessing the cookie from "file://" protocol.

- **Get the value of cookie:**
  request.cookies.key

- **Delete the cookie:**
  response.clearCookie("key");

Cookie-parser - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
      "express": "latest",
      "cookie-parser": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
   console.log("Server started at port 8080");
}

//cookie-parser
var cookieparser = require("cookie-parser");
app.use(cookieparser());

//home
app.get("/", fun1);
function fun1(request, response)
{
   response.cookie("x", "100", { maxAge: 1000 * 60 * 10, httpOnly: true });
   response.send("Cookie created");
}

//getcookie
app.get("/getcookie", fun2);
```

```
function fun2(request, response)
{
   response.send("x is: " + request.cookies.x);
}

//deletecookie
app.get("/deletecookie", fun3);
function fun3(request, response)
{
   response.clearCookie("x");
   response.send("Deleted");
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

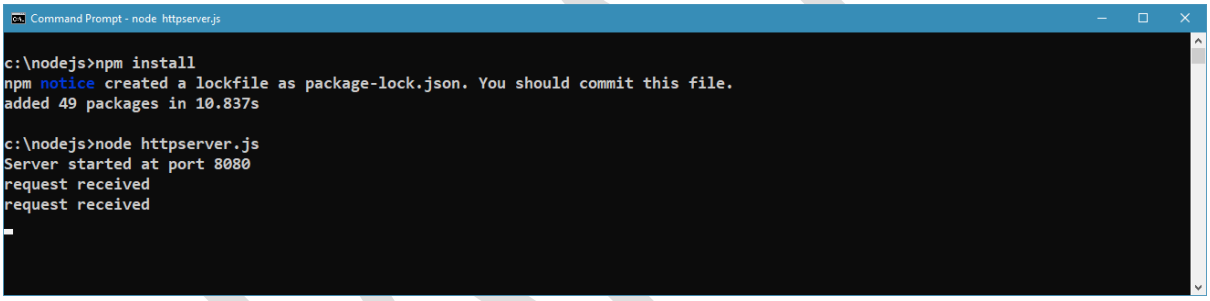
  npm install

  node httpserver.js

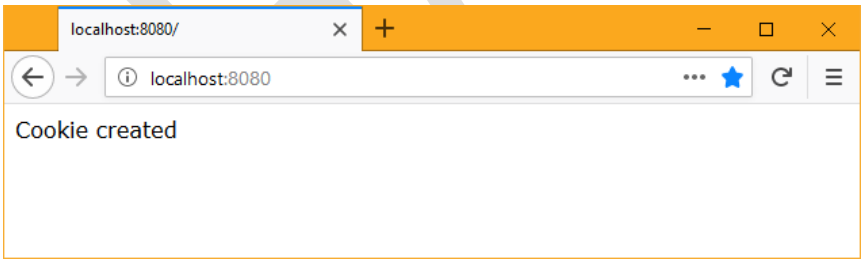
- Open any browser

  http://localhost:8080

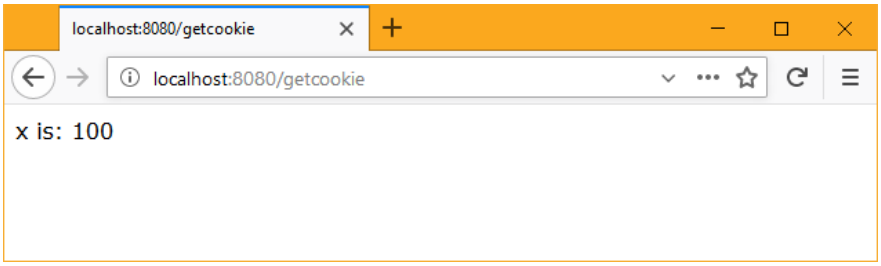
Output:



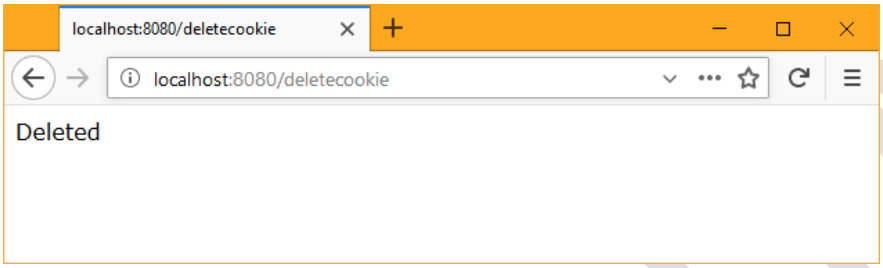
http://localhost:8080



http://localhost:8080/getcookie



http://localhost:8080/deletecookie



express-session Package

- The "express-session" package is used to convert the session data (key-value pairs) into "javascript object".
- Session is a temporary memory location stored at server. For every browser, the server creates a session at server side. The session can store any user-related details of the current browser. The session will be deleted automatically, when the user has closed the browser.

Steps for working with "express-session"

- **Import the "express-session" package in "package.json" file:**

```
"dependencies":
{
    "express-session": "latest"
}
```

- **Acquire "express-session" module:**

```
var expresssession= require("express-session");
```

- **Enable session:**

```
app.use(expresssession( { secret: "secret value here", cookie: { maxAge: milliseconds } } ) );
```

Options:

secret	:	Represents the secret alpha-numeric value, based on which, the session will be encrypted while storing and decrypted while retrieving.
maxAge	:	The lifetime (milli seconds) of cookie that stores session id.

- **Set / get the value of session:**

```
request.session.key
```

- **Delete (clear) the session:**

```
request.session.destroy();
```

Express-session - Example

c:\nodejs\package.json

```
{
```

```
    "name": "mypackage",
    "version": "1.0.0",
    "description": "this is my package",
    "license": "ISC",
    "repository": "none",
    "dependencies":
    {
        "express": "latest",
        "body-parser": "latest",
        "cookie-parser": "latest",
        "express-session": "latest"
    }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
    console.log("Server started at port 8080");
}

//cookieparser
var cookieparser = require("cookie-parser");
app.use(cookieparser());

//expresssession
var expresssession = require("express-session");
app.use(expresssession({secret: "dfgr4h56h567j6767", cookie: { maxAge: 1000 * 60 * 60 } })); //60 min

//bodyparser
var bodyparser = require("body-parser");
app.use(bodyparser.urlencoded({ extended: true }));
app.use(bodyparser.json());

//home
app.get("/", fun1);
function fun1(request, response)
{
    var html = "<form action='/' method='post'>" +
        "Your name: <input type='text' name='username'><br>" +
        "<button type='submit'>Submit</button>" +
        "</form>";
    if (request.session.username)
    {
        html += "<br>Your username from your session is: " + request.session.username;
    }
    response.send(html);
}

//post
app.post("/", fun2);
function fun2(request, response)
```

```
{
    request.session.username = request.body.username;
    response.redirect("/");
}
```

Execution

- Open Command Prompt

  cd c:\nodejs

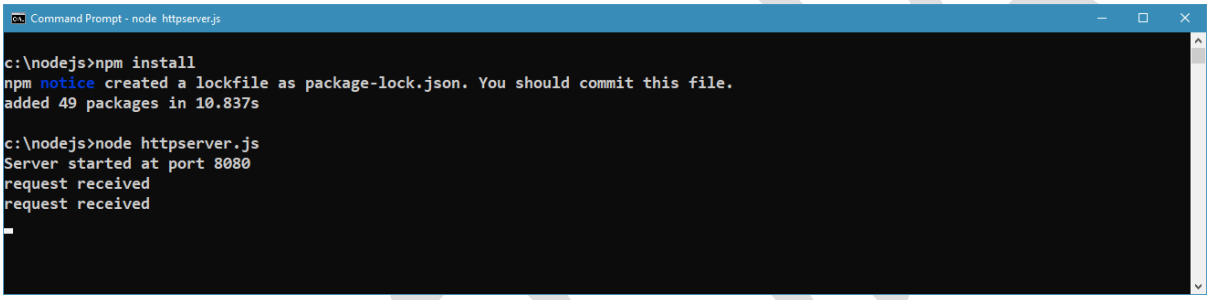
  npm install

  node httpserver.js

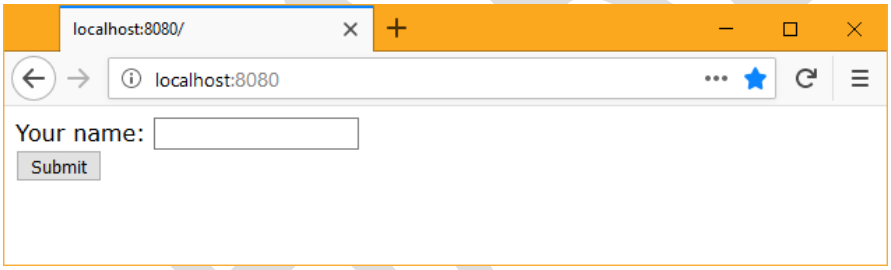
- Open any browser

  http://localhost:8080

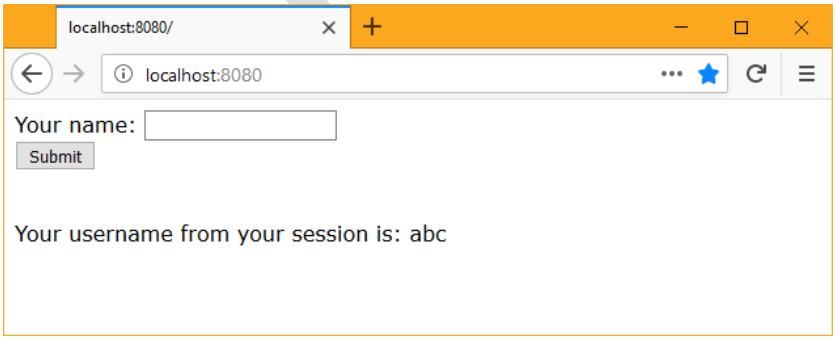
Output:



http://localhost:8080



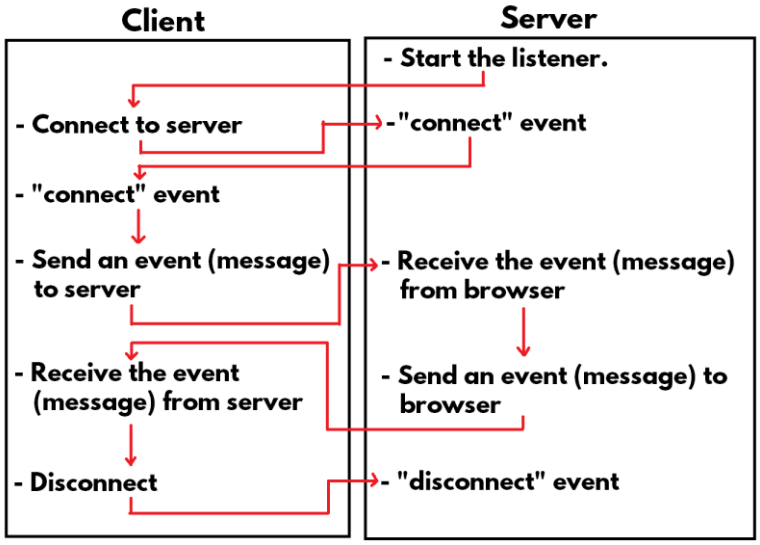
Enter some name and click on "Submit".



Socket.IO

- The "socket.io" package is used to create "chating applications" (messenger applications).

- It provides "client-side API" and "server-side API" for development of chatting applications.



Socket.io – Server side code

- **Import the "socket.io" package in "package.json" file:**

```
"dependencies":
{
   "socket.io": "latest"
}
```

- **Acquire "socket.io" module:**

```
var socketio = require("socket.io");
```

- **Acquire "http" module:**

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

- **Create http server:**

```
var server = http.createServer();
```

- **Create the "socket server" based on "http server":**

```
var io = socketio(server);
```

- **Start the socket server:**

```
server.listen(port no");
```

  - **Define "connect" event:**

```
io.on("connect", function(client)
{
   //client = represents the client (browser)
});
```

Note: The "connect" event executes when the client has connected to the "socket server".

- **Receive event (message) from the client:**

```
client.on("event name", function(data) {
   data = message sent by the client
});
```

- **Send event (message) to specific client:**

```
client.emit("event name", data);
```

- **Send event (message) to all clients:**
  io.sockets.emit("event name", data);

- **Send event (message) to all clients, except the current client:**
  client.broadcast.emit("servermessage", data);

- **Define "disconnect" event:**
  client.on("disconnect", function() {
  });

Socket.io – Client side code

- **Import the "socket.io" client side library file:**
  <script src="/node_modules/socket.io-client/socket.io.js"></script>

- **Connect to server:**
  var server = io.connect("ws://localhost:port");

- **Define "connect" event:**
  server.on("connect", function() {
  });

  The "connect" event executes when the client has connected to the socket server.

- **Send event (message) to server:**
  server.emit("event name", data);

- **Receive event (message) from server:**
  server.on("event name", function(data) {
  //data = message received from server
  });

Socket.io - Example

c:\nodejs\package.json

```
{
  "name": "mypackage",
  "version": "1.0.0",
  "description": "this is my package",
  "license": "ISC",
  "repository": "none",
  "dependencies":
  {
    "express": "latest",
    "socket.io": "latest"
  }
}
```

c:\nodejs\httpserver.js

```
//express
var express = require("express");
var app = express();
app.listen(8080, startup);
function startup()
{
  console.log("Server started at port 8080");
}
```

```
app.use(express.static(__dirname));

//http
var http = require("http");
var server = http.createServer();

//socketio
var socketio = require("socket.io");
var io = socketio(server);
server.listen(7070);

//home
app.get("/", fun1);
function fun1(request, response)
{
    response.sendFile(__dirname + "/index.html");
}

//chat
io.on("connect", fun2);
function fun2(client)
{
    console.log("Client connected");

    client.on("message", fun3);
    function fun3(data)
    {
        console.log(data);
        io.sockets.emit("message", data);
    }
}
```

c:\nodejs\index.html

```
<html>
    <head>
        <title>Socket.IO - Chat</title>
        <style>
            #div1
            {
                width: 100%;
                height: 200px;
                overflow: auto;
                background-color: #84d6e5;
            }
            #txt1
            {
                border: 1px solid blue;
                width: 400px;
            }
        </style>
    </head>
    <body>
        <h1>Socket.IO - Chat</h1>
        <input type="button" value="Connect to server" id="button1">
        <div>
```

```
        Name: <input type="text" id="txt2" value="guest">
    </div>

    <div id="div1">
    </div>
    <div>
        <input type="text" id="txt1" placeholder="Message" autofocus="autofocus">
        <input type="button" id="button2" value="Send">
    </div>

    <script src="/node_modules/socket.io-client/dist/socket.io.js"></script>

    <script>
        var server;
        document.getElementById("button1").addEventListener("click", fun1);
        function fun1()
        {
            server = io.connect("ws://localhost:7070");
            server.on("connect", fun2);
            function fun2()
            {
                document.getElementById("div1").innerHTML += "<div>Connected to server</div>";
                document.getElementById("button1").style.visibility = "hidden";
            }

            server.on("message", fun4);
            function fun4(data)
            {
                document.getElementById("div1").innerHTML += "<div>" + data + "</div>";
            }
        }

        document.getElementById("button2").addEventListener("click", fun3);
        function fun3()
        {
            var s = document.getElementById("txt2").value + ": "+ document.getElementById("txt1").value;
            server.emit("message", s);
            document.getElementById("txt1").value = "";
            document.getElementById("txt1").focus();
        }
    </script>
  </body>
</html>
```

Execution

- Open Command Prompt

  cd c:\nodejs

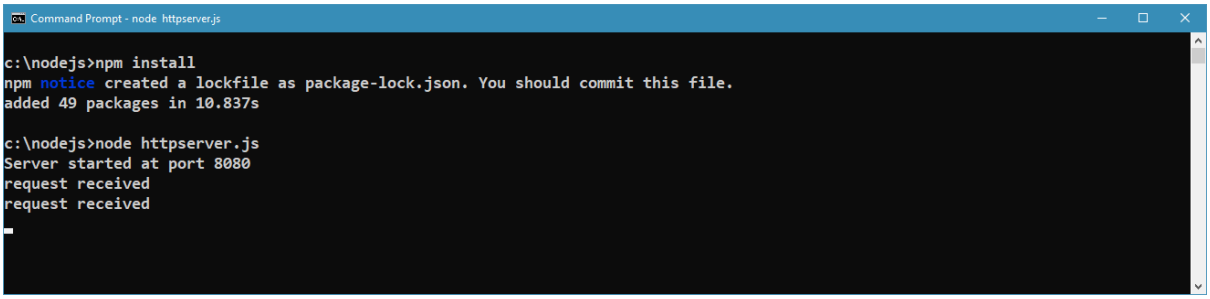
  npm install

  node httpserver.js

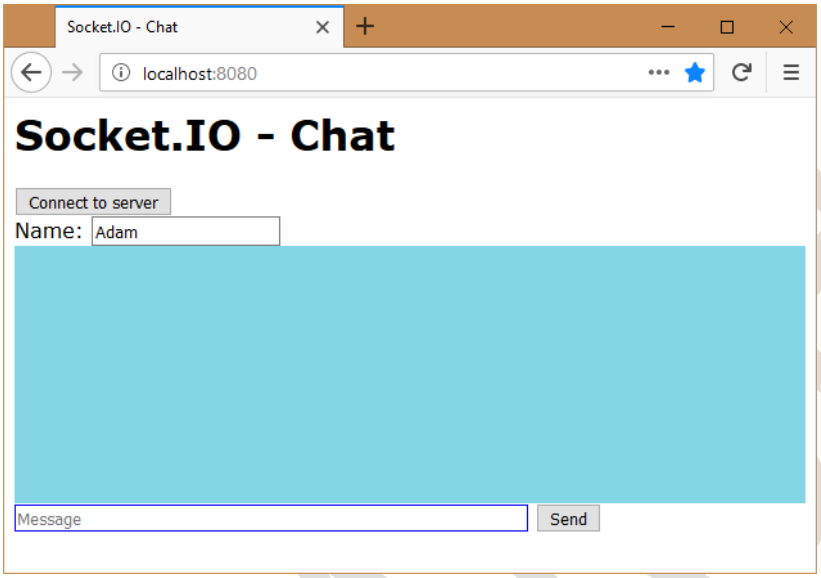
- Open any browser

  http://localhost:8080

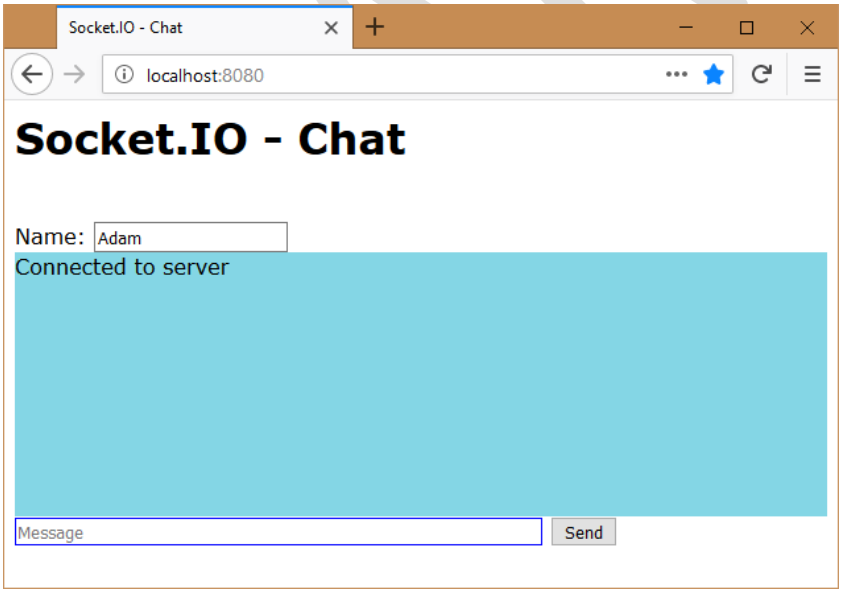
Output:



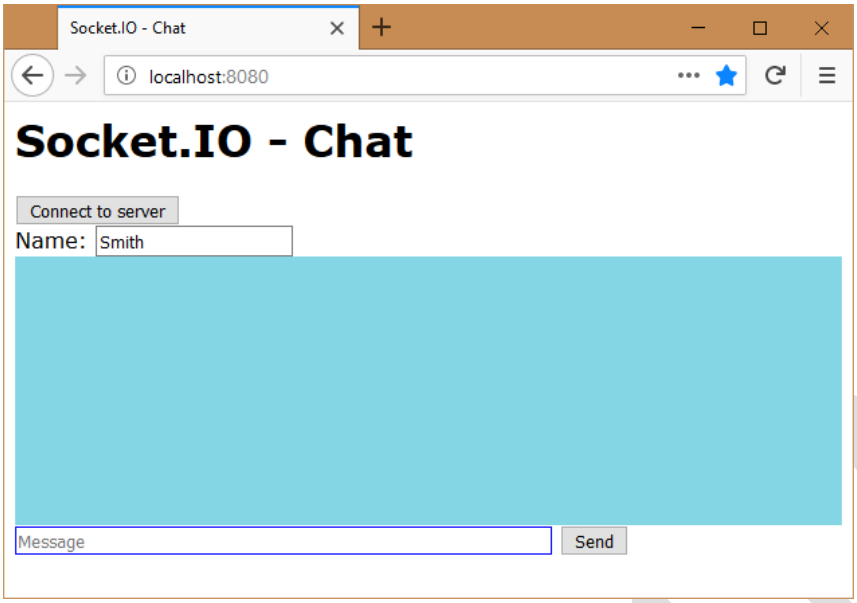
Browser 1: http://localhost:8080



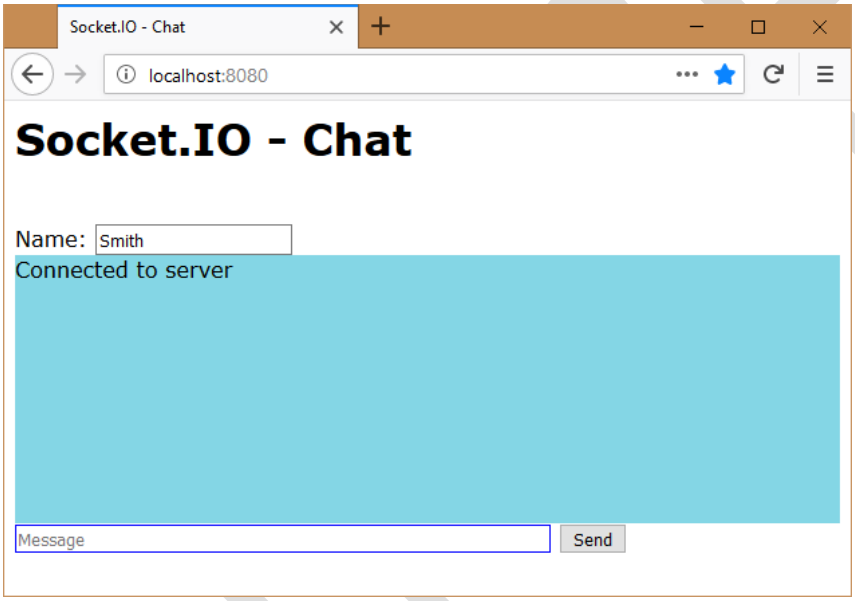
Enter "Adam" and click on "Connect to server".



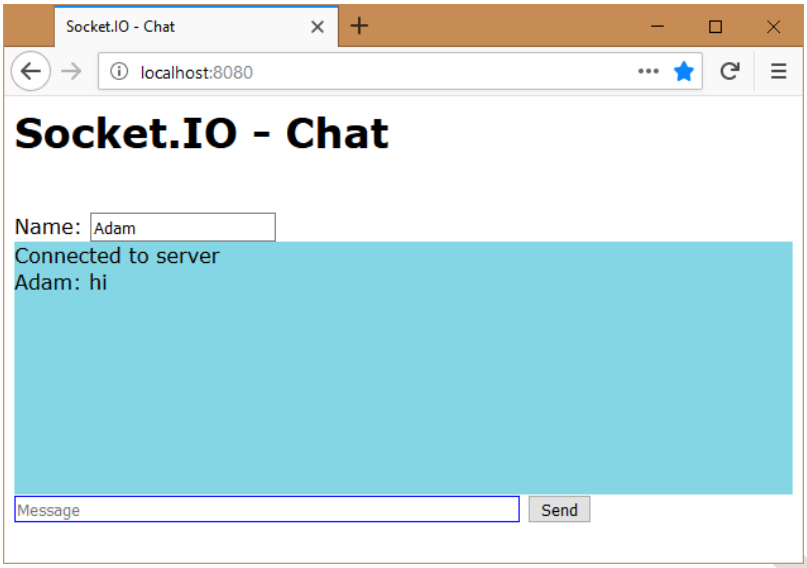
Browser 2: http://localhost:8080



Enter "Smith" and click on "Connect to server".



Browser 1: Enter "hi" message and click on "Send".



Browser 2: Enter "hello" message and click on "Send".