

Install nodejs

\$ sudo apt-get install node

Run nodejs

as interpreter REPL (Read Eval Print Loop) at terminal type : `node`

exit type : `ctrl + c` twice or `ctrl + d`

as compiler using text editor terminal type : `node filename.js`

exit type : `ctrl + c`

run in browser

```
var http = require("http");
http.createServer(function(request, response)
{
    response.writeHead(200,{'Content-Type': 'text/plain'});
    response.end('Hello');
}).listen(8081);
```

`console.log('server running');`

save above code (red) and from terminal enter direktori where file saved.

type : **`node filename.js`**

open browser type : **`http://127.0.0.1:8081/`**

exit type : **`ctrl + c`**

REPL commands

tab key = list of current commands

.help = list of all commands

.break or .clear = exit from multiline expression

.save = save the current node REPL session to a file

.load = load file content in current node REPL session

List down all the locally installed modules

`$ npm ls`

Install node.js module locally:

`$ npm install <module name>`

`$ npm install mongodb` // check in home directory, there is node_modules / mongodb

`$ npm install express` // check in home directory, there is node_modules

nodejs

using module installed in js file

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

Uninstall a Module

```
$ npm uninstall express
```

Updating a Module

```
$ npm update express
```

Search a Module

```
$ npm search express
```

Install globally, add path, we can run it from anywhere

```
$ npm install -g express // -g = global
```

check version (only if installed global)

```
$ node -v
```

```
$ npm -v
```

```
$ mongod --version
```

Blocking code

input.txt

```
nodejs mongodb expressjs
```

vs

```
php mysql codeigniter
```

The program blocks until it reads the file and then only it proceeds to end the program

main.js

```
var fs=require("fs"); // npm install fs
var data=fs.readFileSync('input.txt');
console.log(data.toString());
console.log("Program end");
```

run from terminal

```
$ node main.js
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
nodejs mongodb expressjs
vs
php mysql codeigniter
```

Program end

Non-Blocking Code

input.txt

```
nodejs mongodb expressjs
```

vs

```
php mysql codeigniter
```

The program does not wait for file reading and proceeds to print "End" and at the same time, The program without blocking continues reading the file

main.js

```
var fs=require("fs");
fs.readFile("input.txt", function(err, data)
{
    if(err)
        return console.error(err);
    console.log(data.toString());
});
console.log("End");
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
End
nodejs mongodb expressjs
vs
php mysql codeigniter
```

Even Driven

```
var events=require("events"); // inport module

var eventEmitter = new events.EventEmitter(); // create object eventEmitter
var connectHandler = function connected() // create an event handler
{
    console.log('connection successful');
    eventEmitter.emit('data_received'); // fire the data received event
}

eventEmitter.on('connection', connectHandler); // bind the connection event with the handler
eventEmitter.on('data_received', function() // bind the data received event with the anonymous function
{
    console.log('data received successfully')
});

eventEmitter.emit('connection'); // fire the connection event
console.log("End");
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
connection successful
data received successfully
End
```

Event Emitter

```
var events=require("events"); // inport module

var eventEmitter = new events.EventEmitter(); // create object

var listner1 = function listner1()
{
    console.log('listner1 executed');
}

var listner2 = function listner2()
{
    console.log('listner2 executed');
}

eventEmitter.on('connection', listner1); // bind the connection event with the listner 1 function
eventEmitter.on('connection', listner2); // bind the connection event with the listner 2 function

var eventListeners = require('events').EventEmitter.listenerCount(eventEmitter, 'connection');
console.log(eventListeners + " Listner(s) listening to connection event");

eventEmitter.emit('connection'); // fire the connection event

eventEmitter.removeListener('connection', listner1); // remove the binding of listner 1 function
console.log("Listener1 will not listen now");

eventEmitter.emit('connection'); // fire the connection event

var eventListeners = require('events').EventEmitter.listenerCount(eventEmitter, 'connection');
console.log(eventListeners + "Listner(s) listening to connection event");

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

terminal

```
f@Aspire:~/Documents/node$ node main.js
2 Listner(s) listening to connection event
listner1 executed
listner2 executed
Listener1 will not listen now
listner2 executed
1Listner(s) listening to connection event
End
```

Buffer

Create using integer value / uninitiated buffer (10 octets)

```
var buf = new Buffer(10);
lenbuf = buf.write("nodejs");
console.log(lenbuf);
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
6
```

Create buffer using string

```
var buf = new Buffer("nodejs");
console.log(buf.toString());
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
nodejs
```

Take character start from to

```
var buf = new Buffer("nodejs");
console.log(buf.toString('utf8',0,3));
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
nod
```

Conver buffer to JSON

```
var buf = new Buffer("nodejs");
var json = buf.toJSON();
console.log(json);
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
{ type: 'Buffer', data: [ 110, 111, 100, 101, 106, 115 ] }
```

Concat buffer

```
var x1 = new Buffer("nodejs");
var x2 = new Buffer("javascript");
var x3 = Buffer.concat([x1,x2]);
console.log(x3.toString());
```

terminal

```
f@Aspire:~/Documents/node$ node main.js
nodejsjavascript
```

Compare buffer

```
var x1 = new Buffer("nodejs");
var x2 = new Buffer("node");
var x3 = x1.compare(x2);

if(x3<0)
{ console.log(x1 + " inside " + x2); }
else if(x3==0)
{ console.log(x1 + " same as " + x2); }
else
{ console.log(x1 + " outside " + x2); }
```

template

```
f@Aspire:~/Documents/node$ node main.js
nodejs outside node
```

nodejs

Copy buffer

```
var x1 = new Buffer("nodejs");  
var x2 = new Buffer(6);  
x1.copy(x2);
```

```
console.log(x2.toString());
```

template

```
f@Aspire:~/Documents/node$ node main.js  
nodejs
```

Slide buffer

```
var x1 = new Buffer("nodejs");  
var x2 = x1.slice(1,3);
```

```
console.log(x2.toString())
```

template

```
f@Aspire:~/Documents/node$ node main.js  
nod
```

Length buffer

```
var x1 = new Buffer("nodejs");  
var x2 = x1.length
```

```
console.log(x2);
```

template

```
f@Aspire:~/Documents/node$ node main.js  
6
```

using express

main.js

```
var express = require('express'); //npm install express
var ff=require('body-parser'); //npm install body-parser (handling json, raw, text, url)
var app=express();

app.use(ff.json());

app.get('/',function(req,res)
{
    res.send('mai-mai');
});

app.get('/object',function(req,res)
{
    var obj={name:'Wokki',karir:'jobless programmer'};
    res.send(obj);
});

app.get('/page.html',function(req,res)
{
    res.send('<h1>Hello</h1>');
});

app.listen(8080);
console.log('connected');
```

f@Aspire:~/Documents/node\$ node main.js
connected

type in browser: (if can not change, ctrl+c in terminal then type again node main.js)

localhost:8080



mai-mai

localhost:8080/object



localhost:8080/object/page.html



Hello

insert data to mongodb

make database **stock** from mongodb shell
install at home direktori : \$ npm install mongodb

main.js

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

MongoClient.connect(url, function(err,db)
{
    if(err)
    {
        console.log('Unable to connect',err);    }
    else
    {
        console.log('Connection ok',url);
        var collection = db.collection('shephone'); // collection will automatically created
        var myphone1 = {merk:'mito', type:'a1'};
        var myphone2 = {merk:'advan', type:'a6'};
        collection.insert([myphone1, myphone2],function(err,result)
        {
            if(err)
            {
                console.log(err);    }
            else
            {
                console.log('Insert',result.length, result);    }
            db.close();
        });
    }
});
```

f@Aspire:~/Documents/node\$ node main.js

read data from mongodb

main.js

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

MongoClient.connect(url, function(err,db)
{
    if(err)
    {
        console.log('Unable to connect',err);
    }
    else
    {
        console.log('Connection ok',url);
        var collection = db.collection('shephone');
        collection.find({merk:"mito"}).toArray(function(err,result)
        {
            if(err)
            {
                console.log(err);
            }
            else if(result.length)
            {
                console.log("Found:", result);
            }
            else
            {
                console.log('No document found');
            }
            db.close();
        });
    }
});
```

```
f@Aspire:~/Documents/node$ node main.js
Connection ok mongodb://localhost:27017/stock
Found: [ { merk: 'mito', type: 'a1', _id: 5746cced25def5360e6f535c } ]
```


update data to mongodb

main.js

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

Database : stock

```
MongoClient.connect(url, function(err,db)
```

```
{  
  if(err)  
  {  
    console.log('Unable to connect',err);  
  }  
  else  
  {  
    console.log('Connection ok',url);  
    var collection = db.collection('shephone');  
    collection.update({merk: 'mito'}, {$set: {type: 'a7'}},function(err,numUpdated)  
    {  
      if(err)  
      {  
        console.log(err);  
      }  
      else if(numUpdated)  
      {  
        console.log('Updated succesfully', numUpdated);  
      }  
      else  
      {  
        console.log('No document found with');  
      }  
      db.close();  
    });  
  }  
});
```

Update at merk : mito
Field type change to a7

```
f@Aspire:~/Documents/node$ node main.js
```

read data from mongodb (another way)

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

Database : stock

```
MongoClient.connect(url, function(err,db)
{
    if(err)
    {
        console.log('Unable to connect',err);
    }
    else
    {
        console.log('Connection ok',url);
        var collection = db.collection('myphone'); // find in collection myphone

        var mycursor = collection.find({merk: 'SAMSUNG'});
        mycursor.sort({price:-1}); // sort descending
        mycursor.limit(10); // limit record 10
        mycursor.skip(0); // skip specified record, 0 for skipping 0 record
        mycursor.each(function(err,doc) // iterate result
        {
            if(err)
            {
                console.log(err);
            }
            else
            {
                console.log('view:', doc);
            }
        });
    }
});
```

```
f@Aspire:~/Documents/node$ node main.js
```

```
Connection ok mongodb://localhost:27017/stock
view: { _id: 574565a7a91ef40a166fcab3,
  merk: 'SAMSUNG',
  type: 'Galaxi Tab 3',
  price: 4500000,
  amount: 2 }
view: { _id: 574565a7a91ef40a166fcaaf,
  merk: 'SAMSUNG',
  type: 'A7',
  price: 2300000,
  amount: 4 }
view: { _id: 574565a7a91ef40a166fcab0,
  merk: 'SAMSUNG',
  type: 'J1',
  price: 1500000,
  amount: 10 }
view: null
```

using mongoose to create data

Build API that allow users to CRUD (Create-Read-Update-Delete) using mongoose

install mongoose

```
f@Aspire:~/node_modules$ npm install mongoose
```

create_mongoose.js

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/stock'); //make database stock at mongodb before (if not it will create automatically)
```

This will connect to database stock

//object

```
var LearnSchema = new mongoose.Schema({
  name: String,
  completed: Boolean,
  value: Number,
  updated_at: {type: Date, default: Date.now},
});
```

Schema for object

```
var Learn = mongoose.model('Learn', LearnSchema);
```

Learn / learn / Learns / learns

//new instance

```
var learn = new Learn({name: 'nodejs', completed: false, value:30});
```

New object

```
learn.save(function(err)
```

```
{
  if(err)
    console.log(err);
  else
    console.log(learn);
});
```

Save to database

```
f@Aspire:~/Documents/node$ node create_mongoose.js
{ updated_at: Thu Jun 16 2016 10:51:16 GMT+0800 (WITA),
  _id: 57621424762cf16e0ed3b1ce, // automatically created
  value: 30,
  completed: false,
  name: 'nodejs',
  __v: 0 } // automatically created
```

Check at database: \$mongo
show dbs
use stock
show collections
// there will be learns
db.learns.find() // view data

using mongoose to read data

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/stock');
```

//object

```
var LearnSchema = new mongoose.Schema({
  name: String,
  completed: Boolean,
  value: Number,
  updated_at: {type: Date, default: Date.now},
});
```

```
var Learning = mongoose.model('Learn', LearnSchema);
```

Learn / learn / Learns / learns

```
Learning.find(function(errorr,viewlearn)
```

```
{
  if(errorr) return console.error(errorr);
  console.log(viewlearn)
});
```

Read data from database (view all data)

```
f@Aspire:~/Documents/node$ node find_mongoose.js
[ { updated_at: Thu Jun 16 2016 10:51:16 GMT+0800 (WITA),
  __v: 0,
  _id: 57621424762cf16e0ed3b1ce,
  value: 30,
  completed: false,
  name: 'nodejs' } ]
```

nodejs

using mongoose to read data with more than one criteria

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/stock');
```

database stock

//object

```
var LearnSchema = new mongoose.Schema({
  name: String,
  completed: Boolean,
  value: Number,
  updated_at: {type: Date, default: Date.now},
});
```

```
var Learning = mongoose.model('Learn', LearnSchema);
```

collection Learn / Learns / learn / learns

```
var callback = function(err,data)
{
  if(err) return console.error(err);
  else console.log(data);
}
```

Search data to read at database **stock** collection **learns**
Where field **completed** : **false** and **name** : **express**

```
Learning.find({completed:false , name: /express/}, callback);
```

data from database stock, collection learns

```
> db.learns.find()
{ "name" : "nodejs", "completed" : false, "value" : 30, "_id" : ObjectId("57621424762cf16e0ed3b1ce"), "updated_at" :
ISODate("2016-06-16T02:51:16.610Z"), "__v" : 0 }
{ "name" : "html+css", "completed" : false, "value" : 80, "_id" : ObjectId("57621a536c7cbb750fa419e6"), "updated_at"
: ISODate("2016-06-16T03:17:39.048Z"), "__v" : 0 }
{ "name" : "javascript", "completed" : true, "value" : 70, "_id" : ObjectId("57621a99e634c8810f5c7e93"),
"updated_at" : ISODate("2016-06-16T03:18:49.943Z"), "__v" : 0 }
{ "name" : "express", "completed" : false, "value" : 20, "_id" : ObjectId("57621a99e634c8810f5c7e94"), "updated_at" :
ISODate("2016-06-16T03:18:49.948Z"), "__v" : 0 }
```

f@Aspire:~/Documents/node\$ node find_mongoose.js

```
[ { updated_at: Thu Jun 16 2016 11:18:49 GMT+0800 (WITA),
  __v: 0,
  _id: 57621a99e634c8810f5c7e94,
  value: 20,
  completed: false,
  name: 'express' } ]
```

using mongoose to update data

```

var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/stock');

var LearnSchema = new mongoose.Schema({
  name: String,
  completed: Boolean,
  value: Number,
  updated_at: {type: Date, default: Date.now},
});

var Learning = mongoose.model('Learn', LearnSchema);
Learning.update({completed: false}, {completed: true}, {multi: true}, function(err, success)
{
  if(err) return handleError(err);
  console.log(success);
});

```

database stock

collection Learn / Learns / learn / learns

All document at database **stock** collection **learns**
Where field **completed : false** changed to **completed : true**
multi : true = all documents

```
f@Aspire:~/Documents/node$ node update_mongoose.js
```

using mongoose to update data (use more than one field name)

```

var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/stock');

var LearnSchema = new mongoose.Schema({
  name: String,
  completed: Boolean,
  value: Number,
  updated_at: {type: Date, default: Date.now},
});

var Learning = mongoose.model('Learn', LearnSchema);
Learning.findOneAndUpdate({name: /nodejs/}, {completed: false}, function(err, success)
{
  if(err) return console.log(err);
  else console.log(success);
});

```

Document at database **stock** collection **learns**
Where field **name : nodejs** changed **completed : false**

using mongoose to delete data

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/stock');

var LearnSchema = new mongoose.Schema({
  name: String,
  completed: Boolean,
  value: Number,
  updated_at: {type: Date, default: Date.now},
});

var Learning = mongoose.model('Learn', LearnSchema);
Learning.remove({name: /nodejs/}, function(err, ocee)
{
  if(err) return console.log(err);
  else console.log(ocee);
});
```

Or change remove with
findOneAndRemove

Document at database **stock** collection **learns**
Delete based on field **name** : **nodejs**

Accessing static html & css

[structure directory](#)

myproject (root direktoory)

server.js

public (folder)

 javascripts (folder) → jquery-1.10.1.min.js

 stylesheets (folder) → style.css

 index.html

node_modules (folder)

 mime

 socket.io

package.json

make package.json :

package.json

```
{
  "name": "myproject",
  "version": "0.0.1",
  "description": "simple multiroom chat server",
  "dependencies":
  {
    "socket.io": "~0.9.6",
    "mime": "~1.2.7"
  }
}
```

enter terminal type :

f@Aspire:~/Documents/myproject\$ **npm install**

this will create folder node_modules with **mime** and **socket.io** inside

make server.js :

server.js

```
var http = require('http'); //build http module server & client functionality
var fs = require('fs'); //provides filesystem related functionality
var path = require('path'); //provides filesystem path related functionality
var mime = require('mime'); // add-on mime module provides to derive a mime type based on a filename extension
var cache = {}; // cache object is where the contents of cached files are stored
```

nodejs

//handle the sending of 404 errors when a file is requested that doesn't exist

```
function send404(response)
```

```
{
    response.writeHead(404, {'Content-Type': 'text/plain'});
    response.write('Error 404: resource not found');
    response.end();
}
```

//serves file data, the function first writes the appropriate HTTP headers & then sends the contents of the file

```
function sendFile(response, filePath, fileContents)
```

```
{
    response.writeHead
    (
        200, {"content-type": mime.lookup(path.basename(filePath))}
    );
    response.end(fileContents);
}
```

//determines whether or not a file is cached and if so, serves it.

//if a file is not cached, it's read from disk and served.

//if the file doesn't exist, an HTTP 404 error is returned as a response.

```
function serveStatic(response, cache, absPath)
```

```
{
    if(cache[absPath]) // check if file is cached memory
    { sendFile(response, absPath, cache[absPath]); } // serve file from memory
    else
    {
        fs.exists(absPath, function(exists) // check if file exists
        {
            if(exists)
            {
                fs.readFile(absPath, function(err,data) // read file from disk
                {
                    if(err)
                    {
                        send404(response);
                    }
                    else
                    {
                        cache[absPath] = data; // serve file read from disk
                        sendFile(response, absPath, data);
                    }
                }
            }
            else
            {
                send404(response); } // if file doesn't exists send http 404 response
            }
        });
    }
}
```

//create http server using anonymous function to define per-request behavior

```
var server = http.createServer(function(request, response)
```

```
{
    var filePath = false;
    if(request.url == '/')
    {
        filePath = 'public/index.html'; } // determine html file to be served by default
    else
    {
        filePath = 'public' + request.url; } // translate url path to relative file path
    var absPath = './' + filePath;
    serveStatic(response, cache, absPath); // serve static file
});
```

//start the server 3000 is arbitrary choice

```
server.listen(3000, function()
```

```
{
    console.log("server listening on port 3000");
});
```


nodejs

[make index.html](#)

[index.html](#)

```
<!doctype html>
<html lang="en">

<head>
    <title>HOME</title>
    <link rel="stylesheet" href="/stylesheets/style.css" type="text/css">
</head>
<body>
<div id="content">
    MYPAGE
</div>

<script type="text/javascript" src="/javascripts/jquery-1.10.1.min.js"></script>

</body>
</html>
```

[make style.css](#)

[style.css](#)

```
body
{
    padding: 50px;
    font: 14px "Arial", Helvetica, san-serrif, verdana;
}

#content
{
    width: 800px;
    margin-left: auto;
    margin-right: auto;
}
```

[from terminal type : node server.js](#)

[from browser : http://127.0.0.1:3000](#)

Make Modules

currency.js

```
var canadianDollar = 0.91;
function roundTwoDecimals(amount)
{
    return Math.round(amount * 100)/100;
}

exports.canadianToUS = function(canadian) // can accessed outside modul
{
    return roundTwoDecimals(canadian*canadianDollar);
}

exports.USToCanadian = function(us) // can accessed outside modul
{
    return roundTwoDecimals(us/canadianDollar);
}
```

Module
currency.js

tes-currency.js

Notes : if currency.js is in **subdirectory**
var currency = require('./lib/currency');

```
var currency = require('./currency'); // accessing module currency.js
console.log('50 canadian dollars equals this amount of us dollars:');
console.log(currency.canadianToUS(50));

console.log('30 us dollars equals this amount of canadian dollars:');
console.log(currency.USToCanadian(30));
```

mymodules (Direktory)
- **currency.js**
- **tes-currency.js**

Access and view JSON in browser (just one file.json)

index.html

```

<!doctype html>
<html>
<head></head>
<body>
  <div>sing</div>
</body>
</html>

```

file.json

```

[
  "Javascript is web programming language",
  "Nodejs is server side of Javascript",
  "Mongodb is no-relation database used with Nodejs"
]

```

blog.js

```

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

var server = http.createServer(function(req, res)
{
  getData(res);
}).listen(8000, "127.0.0.1");

function getData(res)
{
  fs.readFile('./file.json', function(err, data)
  {
    if(err)
    {
      hadError(err, res);
    }
    else
    {
      getTemplate(JSON.parse(data.toString()), res);
    }
  })
}

function getTemplate(file, res)
{
  fs.readFile('./index.html', function(err, data)
  {
    if(err)
    {
      hadError(err, res);
    }
    else
    {
      formatHtml(file, data.toString(), res);
    }
  })
}

function formatHtml(file, thetemplate, res)
{
  var thehtml = thetemplate.replace('sing', file.join('<div></div>'));
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end(thehtml);
}

function hadError(err, res)
{
  console.error(err);
  res.end('Server Error');
}

```

Terminal

```
f@Aspire:~/Documents/angular3$ node blog.js
```

Browser

```
http://127.0.0.1:8000
```

```

Javascript is web programming language
Nodejs is server side of Javascript
Mongodb is no-relation database used with Nodejs

```

Event

```
var EventEmitter = require('events'); // built in module
var channel = new EventEmitter();
channel.on('join', function()
{
    console.log('Welcome');
    var a=5;
    var b=4;
    var c=a+b;
    for(var x=0; x<=c; x++)
    {
        console.log(x);
    }
});
channel.emit("join");
```

nodejs

create node project

f@Aspire:~\$ `npm install live-server` // install live server, cek home/node_modules

create directory : `test`

f@Aspire:~/Documents/test\$ `npm init`

```
name: (test) test
version: (1.0.0)
description: only test
entry point: (index.js)
test command: echo ok
git repository:
keywords: mao mao
author: wokilab
license: (ISC) ISC
About to write to /home/f/Documents/test/package.json:
```

```
{
  "name": "test",
  "version": "1.0.0",
  "description": "only test",
  "main": "index.js",
  "scripts": {
    "test": "echo ok"
  },
  "keywords": [
    "mao",
    "mao"
  ],
  "author": "wokilab",
  "license": "ISC"
}
```

Is this ok? (yes) yes

check directory test, there are file : `package.json`

f@Aspire:~/Documents/test\$ `npm i angular2 -S` // install angular2 from npm, check directory test
//parameter i stands for install and S for save will add the installed library to package.json

open package.json and edit :

```
"scripts": {
  "start": "live-server --port=12345" // start live server
},
```