

# NodeJS - Tutorial

Install NodeJS: <https://nodejs.org/en/>

## NPM (<https://www.npmjs.com>)

NPM is the package manager for NodeJS, it comes prepackaged with NodeJS, so when you install NodeJS on your System, you already have NPM.

To initialise what file we want to listen to use "npm init" and follow the guide.

```
Mandatory1
forneus@forneuss-MacBook-Pro ~/Dropbox/Datamatiker/4. Semester/NodeJS/Mand
datory1$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (Mandatory1)
Sorry, name can no longer contain capital letters.
name: (Mandatory1) package.json
version: (1.0.0)
description: This is our test package
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC) GIT
Sorry, license should be a valid SPDX license expression (without "LicenseRe
f"), "UNLICENSED", or "SEE LICENSE IN <filename>".
license: (ISC) MIT
About to write to /Users/forneus/Dropbox/Datamatiker/4. Semester/NodeJS/Mand
atory1/package.json:

{
  "name": "package.json",
  "version": "1.0.0",
  "description": "This is our test package",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "MIT"
}

Is this ok? (yes) y
```

IMPORTANT that you include dependencies in your package.json file. Should look something like this:

```
{
  "author": "Fredrik Bjorklund",
  "description": "Tutorial Package",
  "version": "1.0.0",
  "license": "MIT",

  "dependencies": {
    "express": "4.14.1"
  }
}
```

When we have a proper Package.json file we can use "sudo npm install". This installs all the packages into our "node\_modules" folder.

## Interesting packages:

Nodemon (<https://nodemon.io>)

Nodemon listens to our server and automatically restarts the server if there have been any changes to our server source code. First you have to install nodemon using this npm-command:

```
sudo npm install -g nodemon
```

After that we can use nodemon "our-server-filename-here" as such:

```
odeJS/Mandatory1 ➤ nodemon server.js
[nodemon] 1.11.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node server.js`
```

Budo (<https://www.npmjs.com/package/budo>)

We use this so that we can just look at our webpage without reloading everything. It listens to changes in our designated file, for example "index.html". It can also listen to " \*.js " files.

Budo's default port is 9966. So to go to our locally hosted site we can use:

- <http://localhost:9966>
- 192.168.0.\*\*\*:9966/

To indicate what file to listen to use "budo --live \*\*\*\*.html"

Example:

```
x forneus@forneuss-MacBook-Pro ~/Dropbox/Datamatiker/4. Semeste  
andatory1/view budo --live index.html  
[0000] info Server running at http://192.168.0.15:9966/ (connect)  
[0000] info LiveReload running on 35729
```

## ExpressJS

ExpressJS is a framework for Node.JS. It's used to create Web-Applications.

To import our ExpressJS module we have to start our JS-file with:

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

Then we have to define our Application:

```
Var app = express();
```

If we want to keep our application private we can also use:

```
app.disable('x-powered-by');
```

See below screenshot with comments:

```

1  // A NodeJS server using express framework
2
3  // We need this to use express
4  var express = require("express");
5  // This is the constructor for express
6  var app = express();
7  |
8  var fs = require("fs");
9
10 //We point to the Root using /. This is our landing page.
11 app.get("/", function( oReq, oRes)
12 {
13     //We tell express where our Index.html file is and store it in sIndex...
14     var sIndexHtml = fs.readFileSync("view/index.html", "utf8");
15
16     var sNick = "Forneus Brandt";
17
18     // This replaces the Username with var sNick{}
19     sIndexHtml = sIndexHtml.replace("{{sNick}}",sNick);
20     // Prints it to console
21     console.log(sIndexHtml);
22
23     // Responds with user name
24     oRes.send(sIndexHtml)
25
26 });
27
28 // This is where we make our server.
29 app.listen(8080, function(){
30     console.log("Server is listening on port 8080");
31 });
32
33 // Here we create a second server. We can create as many as we like.
34 app.listen(9090, function(){
35     console.log("Server2 is listening on port 9090");
36 });

```

We can structure our URLs, by passing either a Query string or by using params, see below example:

```

4   var aNicknames = [];
5
6   //app.post has unlimited URL size
7   //http://localhost:1250
8   // app.get("/", function(req, res)
9   // {
10  //   var sNickname = req.query.nickname;
11  //   var secretMessage = req.query.secretmessage;
12  //   //http://localhost:1250/?nickname=Killer
13  //   //http://localhost:1250/?nickname=Killer&secretmessage=FUCKERSDIE
14  //   res.send("Welcome " + sNickname + " " +secretMessage);
15  // });
16
17
18  // We will not get a new way of JQuery with / instad of ,?, &.
19  // game.com/new-user/KILLER
20  // game.com/new-user/MEGAKILLER
21  // url could also be written: /new-user/:nickname/:message
22  // this url gives the same result.
23  // http://localhost:1250/new-user/Killer/message/hello
24  // app.get("/new-user/:nickname/message/:message", function(req,res)
25  // {
26  //   // now instead of query we use params
27  //   var sNickname = req.params.nickname;
28  //   var sMessage = req.params.message;
29  //   res.send(sNickname+" says: "+sMessage);
30  // });

```

## JSON

This is a JSON Object. An empty json is just {}. JSON is constructed as key value pairs as seen below.

```

40   // JSON
41
42   var jPerson =
43   {
44     "firstName" : "John",
45     "lastName" : "Doe",
46     "children" : [],
47   }

```

# SSH to Server

In terminal type: `ssh "username"@xxx.xxx.xxx.xxx`

User for NodeJS: user63

Password: a

IP: 217.61.218.68

Port:10063

PortNumber: 10063

```
forneus@forneuss-MacBook-Pro ~ % ssh user63@217.61.218.68
The authenticity of host '217.61.218.68 (217.61.218.68)' can't be established.
ECDSA key fingerprint is SHA256:CfLsWjuv0Ld3DZX5pT+LOYQXysGVpRiYRms4K1bre7I.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '217.61.218.68' (ECDSA) to the list of known hosts.
user63@217.61.218.68's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-41-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

27 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

user63@server:~$
```

## QUOTA / SPACE IN THE SERVER

This defines how much HDD space that has been allocated for a specific user on a Server, see below:

```
user63@server:~$ quota
```

```
Disk quotas for user user63 (uid 1054): none
```

## FLEXBOX

FlexBox is used to design the Layout of our webpage. No more Tables, Blocks etc.

A good way to train and sharpen your flexbox skills is to play som FlexBox Froggy:

<http://flexboxfroggy.com/>

Flex box i structured by having a Flex-Container within which we have a flex-line on which our Flex-items are ordered.

HTML:

```
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
</div>
```

CSS:

```
.flex-container {
  display: -webkit-flex;
  display: flex;
  width: 300px;
  height: 240px;
  background-color: Silver;
}

.flex-item {
  background-color: DeepSkyBlue;
  width: 100px;
  height: 100px;
  margin: 5px;
```

Result:



# HTML

HTML is Hyper Text Markup Language. It's the code that's interpreted by our browsers to represent a site.

# CSS

CSS is used to style our HTML code. This can be done directly in HTML or implemented by using a separate CSS stylesheet. There are 2 ways that we can access our HTML items. One is by accessing through ID by using the dot operator '.':

```
.center {  
  text-align: center;  
  color: red;  
}
```

Or a HTML Class by using '#'

```
#para1 {  
  text-align: center;  
  color: red;  
}
```

Ideally you want to use Class for a group of HTML items that have some common attributes. ID will be used to identify unique items and in CSS have a unique styling. If we want to select all elements of a specific type we can just write for example: `<h1>{*DO CSS*};`



# SFTP - FTP

FTP is short for file-transfer-protocol.

SFTP is short for secure-file-transfer-protocol.

FTP is easily read by humans but it's not as secure as SFTP. SFTP is more complex and doesn't require a continuous connection between server-client. Generally speaking SFTP is superior to FTP, but FTP has it's merits. It's widely known/used/accepted. Personally i'll primarily focus on SFTP considering security, which is ever more important to WEB.

# PORTS

Already know & understand, unnecessary for my guide.

# FOREVER

<https://www.npmjs.com/package/forever>

Forever is part of npm. This is what we use to launch a NodeJS server literally forever.

Command to install forever is "npm install forever"

Command to run forever on a server "forever start \*serverfile\*.js"

Command to kill/stop our server "forever stopall"

Command to restart a server "forever restart \*serverfile\*.js"

It very much functions the same as Nodemon except it's not dependant on our terminal running to keep the server live.

# REST

Source: [http://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)

I have to ask you about this as to how this is implemented in NodeJS.

# SYNTACTICAL FOOBARS

`indexOf` // Used to grab a certain element from an Array at index position e.g. `[4]` for grabbing index 4 which is position 5, since arrays are 0-indexed.

`array.includes()` // This will automatically iterate through an array and return a Boolean value if a given element is contained in the array. It will stop after the first hit, thus not iterating through the entire array.

`res.json`

Sends JSON response.

`res.sendFile`

`app.get("/", function(req, res)`

`{`

`res.sendFile(__dirname + "/index.html");`

`});`

This sends a file depending on params request.

JQUERY select id # class .

// Cover later

Jquery empty

`jQuery.isEmptyObject({}); // true`

`jQuery.isEmptyObject({ foo: "bar" }); // false`

Jquery append

// cover later

Jquery text

// cover later

Jquery html

// cover later