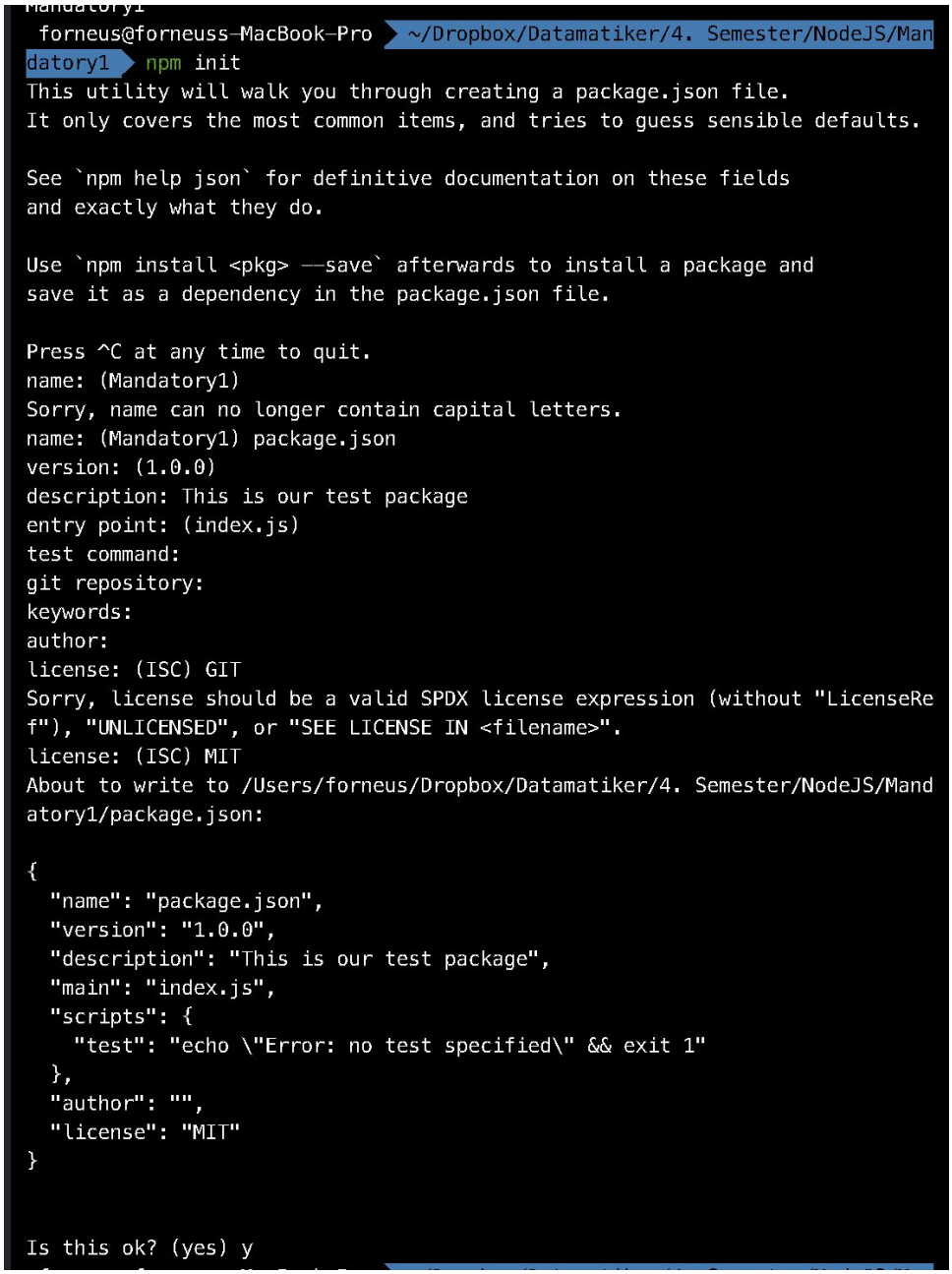
NodeJS - Tutorial

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.



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.

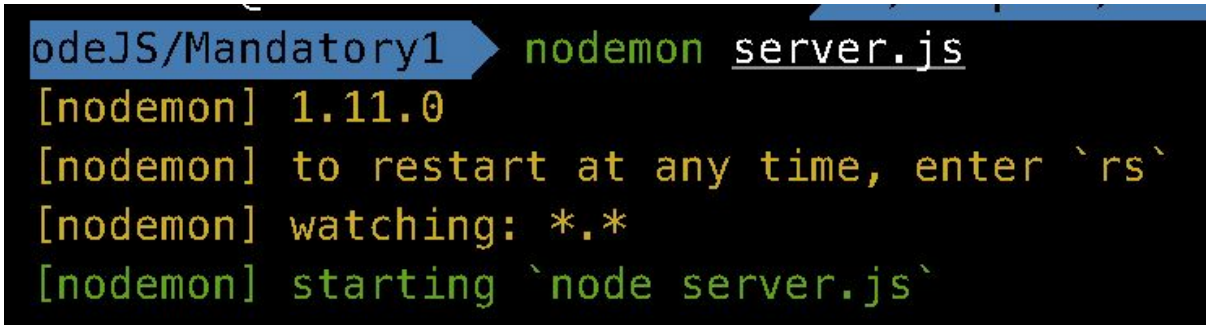
Interesting packages:

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:



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:

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:

```
// A NodeJS server using express framework

// We need this to use express
var express = require("express");
// This is the constructor for express
var app = express();

var fs = require("fs");

//We point to the Root using /. This is our landing page.
app.get("/", function( oReq, oRes)
{
  //We tell express where our Index.html file is and store it in sIndex...
  var sIndexHtml = fs.readFileSync("view/index.html", "utf8");

  var sNick = "Forneus Brandt";

  // This replaces the Username with var sNick{}
  sIndexHtml = sIndexHtml.replace("{{sNick}}",sNick);
  // Prints it to console
  console.log(sIndexHtml);

  // Responds with user name
  oRes.send(sIndexHtml)

});

// This is where we make our server.
app.listen(8080, function(){
  console.log("Server is listening on port 8080");
});

// Here we create a second server. We can create as many as we like.
app.listen(9090, function(){
  console.log("Server2 is listening on port 9090");
});
```

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

```
var aNicknames = [];

//app.post has unlimited URL size
//http://localhost:1250
// app.get("/", function(req, res)
// {
//   var sNickname = req.query.nickname;
//   var secretMessage = req.query.secretmessage;
//   //http://localhost:1250/?nickname=Killer
//   //http://localhost:1250/?nickname=Killer&secretmessage=FUCKERSDIE
//   res.send("Welcome " + sNickname + " " +secretMessage);
// });


// We will not get a new way of JQuery with / instad of ,?, &.
// game.com/new-user/KILLER
// game.com/new-user/MEGAKILLER
// url could also be written: /new-user/:nickname/:message
// this url gives the same result.
// http://localhost:1250/new-user/Killer/message/hello
// app.get("/new-user/:nickname/message/:message", function(req,res)
// {
//   // now instead of query we use params
//   var sNickname = req.params.nickname;
//   var sMessage = req.params.message;
//   res.send(sNickname+" says: "+sMessage);
// });
```

JSON

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

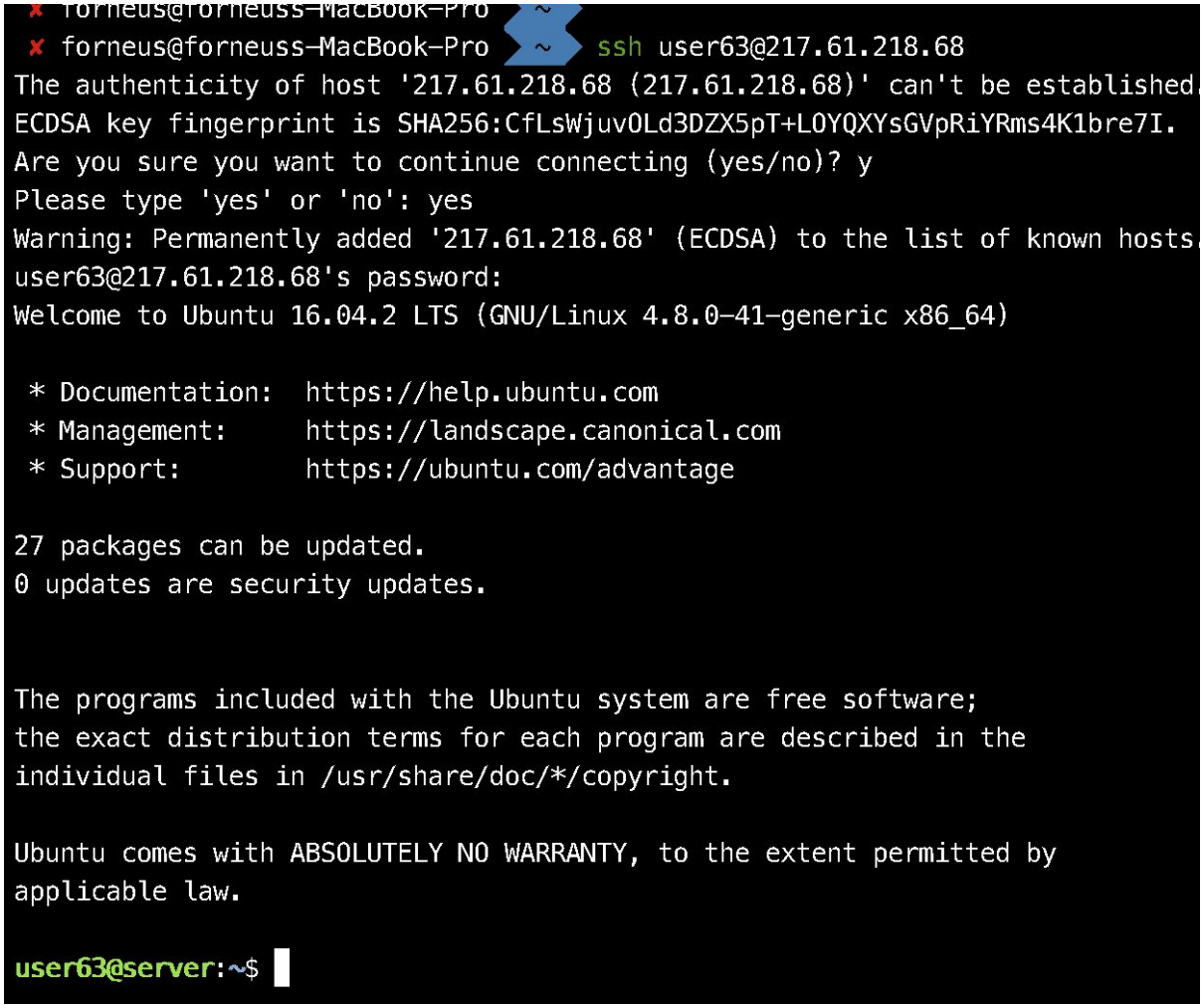
```
// JSON

var jPerson =
{
  "firstName" : "John",
  "lastName" : "Doe",
  "children" : [],
}
```

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



QUOTA / SPACE IN THE SERVER

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

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 seperate CSS stylesheet. There is 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 exmaple: <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
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