

Checklist for node and npm

Purpose:

The following document has been prepared for the development of node applications using npm. In the checklist you will find how to do the following:

Checklist:

1. Initialize an npm project
2. Install and save dependencies within the project
3. Use dependencies within the project
4. How to initialize someone else's project

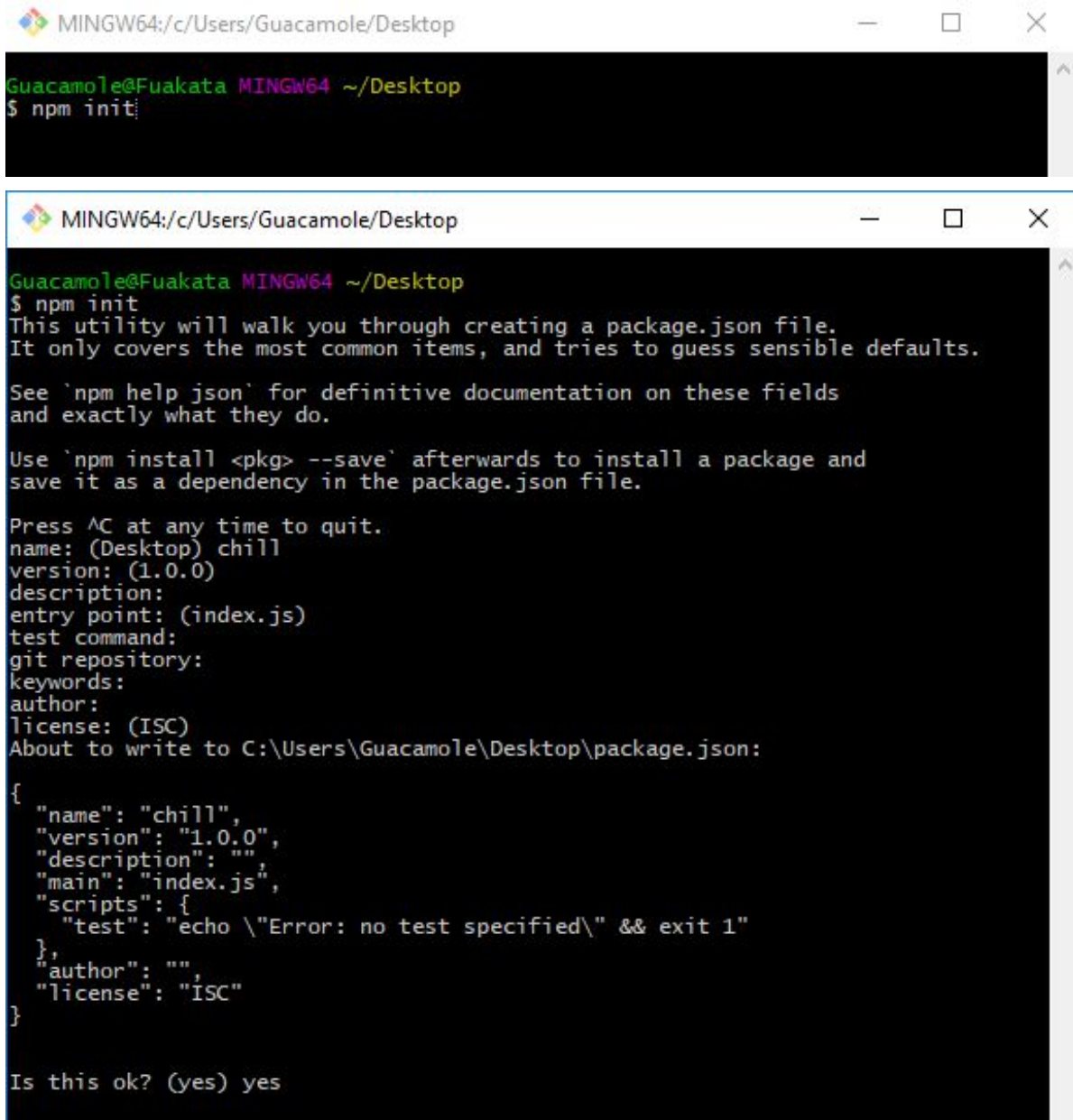
Steps:

1.)

In order to initialize an npm project you must first be located within the desired directory.

Secondly, you must type the following command within the terminal to create a package.json file in the root of the project folder:

\$ npm init



The image shows two screenshots of a Windows command prompt window. The title bar indicates the path is 'MINGW64:/c/Users/Guacamole/Desktop'. The first screenshot shows the command '\$ npm init' being entered. The second screenshot shows the output of the command, which is an interactive prompt for creating a package.json file. The prompt asks for a name, version, description, entry point, test command, git repository, keywords, author, and license. The user has entered 'chill' for the name, '1.0.0' for the version, and 'ISC' for the license. The prompt also shows the generated package.json file content.

```
MINGW64:/c/Users/Guacamole/Desktop

Guacamole@Fuakata MINGW64 ~/Desktop
$ npm init

Guacamole@Fuakata MINGW64 ~/Desktop
$ 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: (Desktop) chill
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\Guacamole\Desktop\package.json:
{
  "name": "chill",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this ok? (yes) yes
```

You will be prompt to name and write each aspect of the package.json file with questions. You can skip the questions manually by hammering the enter key. If you would like to skip this part of the initialization of the package.json file by running the following command:

\$ npm init -y

The package.json file that you create with **npm init** allows npm (node package manager) to track project dependencies as well as track developmental dependencies in the project.

```

1  {
2    "name": "chill",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "ISC"
11 }
12

```

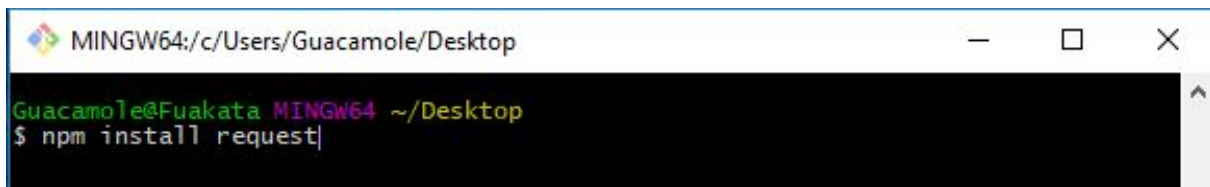
2.)

Once the package.json has been created in the root of your project's folder, you are now able to download and track packages for your project. In order to install packages for your project's use you need to run the following command in any part of the folder where your project is located:

\$ npm install {name of package}

OR

\$ npm install {names of packages}



```

MINGW64:/c/Users/Guacamole/Desktop
Guacamole@Fuakata MINGW64 ~/Desktop
$ npm install request

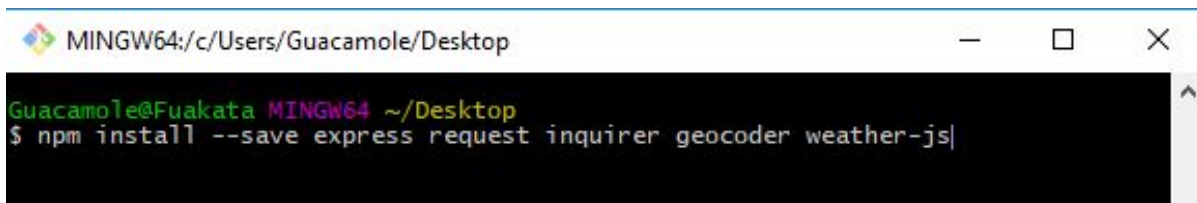
```

Now, this is a command that allows you to successfully install and use packages for your project. However, your project will be poorly made unless you track what the specific dependencies are in your project using your package.json file. In order to properly keep track of your project's dependencies you must use the **--save** option alongside the previous command stated above, exemplified below:

\$ npm install --save {name(s) of package(s)}

OR

\$ npm install {name(s) of package(s)} --save



```

MINGW64:/c/Users/Guacamole/Desktop
Guacamole@Fuakata MINGW64 ~/Desktop
$ npm install --save express request inquirer geocoder weather-js

```

By using the option **--save**, you will be able to save the specific version of your dependencies. The reason it would be useful to keep track of the version of the dependencies of the packages you use is to avoid using versions of the package that may have deprecated (dropped the use of) certain function names in the package that your project depends on.

3.)

In order to use the dependencies that have now been both installed and tracked inside of your package.json file, you must import them into the code that is meant to use those packages using the following syntax in your javascript file:

const {name of package as stated in the documentation of the package} = require("{name of package}");

If done successfully, you will now be able to use the sample code provided within the documentation. An example is provided below with the package called request:

// REQUIRING/ IMPORTING PACKAGE REQUEST

```
const request = require('request');
```

// USING REQUEST AS PER SHOWN WITHIN THE DOCUMENTATION IN [NPMJS.COM](https://npmjs.com)

```
request('http://www.google.com', function (error, response, body) {  
  console.log('error:', error); // Print the error if one occurred  
  console.log('statusCode:', response && response.statusCode); //  
  Print the response status code if a response was received  
  console.log('body:', body); // Print the HTML for the Google  
  homepage.  
});
```

```
1 // REQUESTING THE PACKAGE REQUEST  
2 const request = require("request");  
3  
4 // USING SAMPLE CODE FROM THE REQUEST DOCUMENTATION  
5 request('http://www.google.com', function (error, response, body) {  
6   console.log('error:', error); // Print the error if one occurred  
7   console.log('statusCode:', response && response.statusCode); // Print the response status code if a response was received  
8   console.log('body:', body); // Print the HTML for the Google homepage.  
9 });
```

4.)

Now that we have learned how to initialize a package.json, how to save and track the version of the dependencies for our project, and learned how to incorporate the code into our project you should be aware that you can also use this knowledge to clone and play other people's public repos in a similar fashion. This is slightly different and simpler than the three steps above as most of the work has already been done for you. The workflow is actually just as described below.

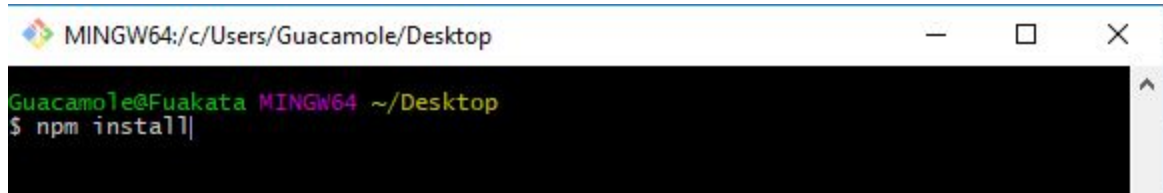
Clone project using the terminal command:

\$ git clone {url of project}

AND THEN

Download all the dependencies of the project with the terminal command:

\$ npm install

A screenshot of a Windows terminal window. The title bar at the top reads 'MINGW64:/c/Users/Guacamole/Desktop' and includes standard minimize, maximize, and close buttons. The terminal content shows a green prompt 'Guacamole@Fuakata', a pink 'MINGW64' environment indicator, and a yellow path '~ /Desktop'. Below this, the command '\$ npm install' is entered in white text on a black background. A vertical scrollbar is visible on the right side of the terminal pane.

```
MINGW64:/c/Users/Guacamole/Desktop  
Guacamole@Fuakata MINGW64 ~/Desktop  
$ npm install
```