NodeJS Cheatsheet

Getting Started

> Steps for getting yourself up and running with a Node application

Keep this open for reference as to what you're doing: Using a package.json

1. Create your directory/repository for your application and `cd` into it
2. Now you need to initialize your application. By doing so, you are going to be creating a `package.json` file that serves as an application manifest that the computer running this application can use to figure out what it needs to have in order for it to run correctly. To get this application started, type in the following into your command line and fill out the questionnaire that follows:

```
npm init
```

1. Once your `package.json` is created, you now need to think about what your application needs in order to run. Usually this consists of finding NPM packages or libraries to import into your application (i.e. using the `request` package). So you will need to do some Googling or go right to npmjs.com and find packages you'd like to use.

==================================

Installing Packages

> Your app is initialized, time to start bringing in some libraries to make your coding go more smoothly

1. You've found the libraries/packages you want to use to support your application's functionality, it's time to install them. You can do this by typing into your command line the following command

```
npm install nameOfPackage
```

1. (cont) This adds that package as a `dependency` of your application and now you can use it in your code as you see fit (see that package's documentation for useage examples).If you want to install multiple packages at once, just keep listing package names with a space between`npm install nameOfPackage1 nameOfpackage2 etc`

2. Now that you're packages are installed, it's time to start coding! First thing you'll want to do is come up with a general layout and file structure for your application. This typically involves creating an `entry point` for your application (often named as `server.js` or `app.js`)
3. To include the packages you've installed you can use Node's built in `require()` to import it and store it in a variable For NPM packages or built in Node packages: `var nameOfPackage =`

```
require("nameOfPackage");
```

> For importing data/functionality from another JS file you've created module.exports:
>
> ```
> var nameOfData = require("./path/to/file.js");
> ```

1. To `export` any data to be made importable by other files, you must assign that data to be part of that file's `module.exports` object.

```
module.exports = {
  data: valueOfData,
  functionName: nameOfFunction,
  etc: valueOfEtc
}
```

====================================

Installing an existing application

> Because you won't always be starting from scratch.

1. Let's say someone sends you a link to their application and you clone it to your computer to use it locally or build upon it. You don't need to look at the files and guess what dependencies are needed (this is too hard / not maintainable) since this application will have come with a `package.json` file. This file has a list of all of the dependencies, and NPM is smart enough to look here and do all of the "setup" work for you. All you need to do is type the following into your command line and you will be all set.

```
npm install
```

1. Now that the dependencies are installed, you can now add/edit/run this application as you see fit.