

# Classes and Objects | `this` keyword | `new` keyword | function constructors

---

- Notes By [Milind Mishra](#)
- Github Repo: [005\\_oops\\_in\\_js](#)

## Making a NPM Package

- Create a folder with the name of the package
- Create a `package.json` file with `npm init` command
- Create a `index.js` file with the code

Preparing the package for publishing to NPM. Here is the structure of the package

```
algorithms  datastructures  index.js  package.json
```

- `index.js` is the entry point of the package
- `package.json` contains the meta data of the package
- folders `algorithms` and `datastructures` contains the code of the package

The `index.js` file

```
// Code Snippets for Data Structures and Algorithms
const DataStructures = require('./dsasnippets/datastructures');
const Algorithms = require('./dsasnippets/algorithms');

// Export the DataStructures and Algorithms modules
module.exports = {
  DataStructures,
  Algorithms
}
```

The folder consists of the code that is exported in the `index.js` file

```
algorithms  datastructures
```

The `algorithms` folder consists of the code for the algorithms. Lets say we have the folder structure as follows

```
algorithms
├─ binarysearch.js
```

```
├─ bubblesort.js
├─ insertionsort.js
├─ linearsearch.js
├─ mergesort.js
├─ quicksort.js
├─ selectionsort.js
└─ utils.js
```

The **datastructures** folder consists of the code for the data structures. Lets say we have the folder structure as follows

```
datastructures
├─ binarysearchtree.js
├─ graph.js
├─ linkedlist.js
├─ queue.js
├─ stack.js
└─ utils.js
```

The individual folder can have a `index.js` file that has the named exports for the files in the folder

- You can structure and publish the package as you like on npm.js website, following the tutorial at <https://docs.npmjs.com/creating-node-js-modules> and the npm package at <https://docs.npmjs.com/creating-and-publishing-scoped-public-packages>.

## Using the package

- Install the package using `npm install <package-name>`
- Import the package in the code using `const <package-name> = require('<package-name>')`
- Use the package in the code using `<package-name>.<module-name>.<function-name>`
- Example

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

// Use the DataStructures module
const { DataStructures } = dsasnippets;

// Use the Algorithms module
const { Algorithms } = dsasnippets;
```