NPM - Modules

Overview

The node package manager (npm) registry contains many packages, which can be installed by a user and many of which are node modules which can be imported by programs and used. A package is a file or directory described by its package.json file and once created, they can be published as private or public. To publish packages you must be signed up as a npm user or organisation.

Package Scope

The scope of a package is either scoped or unscoped.

Unscoped packages are referenced generally, meaning the package name must be unique to all other packages available on the registry. Unscoped packages are always publicly available, and can be referenced simply by package name.

Scoped means it is specific to one user and since usernames are unique, it can therefore have the same name as other packages. To reference a scoped package use the following syntax:

@<scope>/<package-name>

Since scoped packages belong to a user or organisation, they are private by default. Scoped packages can be made public using CLI flags on publishing.

Creating a package.json

The package.json contains the information required for the package to be published on the public registry. A typical file will contain:

* General details
* Tests
* Dependencies
* Versioning rules
* Information to make package reproducible

There are two fields in a package.json which are required and another optional:

* name (required) - lowercase package name, must be one word but can include hyphens and underscores
* version (required) - semantic version number in 'x.x.x' form
* author (optional) - name and optional email and website in the following form (Your Name <email@example.com> (http://example.com) )

To create an initial template package.json, run the following command in the package repository:

npm init

Then follow the template instructions to complete.

Many additional fields can also be added:

* Keywords - aid user search by adding keywords which relate to the project in array of strings format, e.g "keywords": ["erb", "rails" ]
* Engines - versions of node the package will work on, e.g. "engines": { "node": ">=0.10.0" }
* Contributors - give credit where its due, e.g. "contributors": [ 'Chris' ]

Create a Module

Node modules use the CommonJS format of modules, which involves module.exports and require. In a npm package, the main access point is the index.js file, and within that functions and values are added to the exports property of module. For example,

// module.js

module.exports = {

getName: () => {

return 'Chris'

}

}

Will add the function getName to exports, and it can then be called from another script using:

// anotherScript.js

const { getName } = require('./module);

console.log(getName())

The system of adding functions to module.exports is common in packages.

Publishing the Module

Ensure the package version number is correct. Login to npm by following:

npm login

Publish the package unscoped and publically using:

npm publish

Importing the Module

Like any package on the npm registry, there are two popular package managers used to download and install packages to projects, npm, and Yarn:

npm install <package-name>

yarn add <package-name>

Using the Module

Use a module by requiring its module name (folder name) in a script. Depending on the module being imported, the whole export can be required, or just specific functions and properties. For example:

const erbTransformer = require('erb-transformer')

const { name, dob } = require('person')

Node will then search its module.paths locations for either the moduleName.js file or the directory name of the module, then resolve the module file using either index.js or the package.json entry point if present.