4. はじめての npm

npm とは

npm(Node Package Manager) は、Node.js のパッケージ管理ツールです。最新の JavaScript 開発 は npm や yarn といったパッケージ管理ツールを利用するのがスタンダードのため、慣れておきましょう。

パッケージ（package）とは

パッケージは、プログラムでよく利用する便利な機能をまとめたものです。例えば Node.js の MVCフレームワーク Express や、モジュールハンドラツールの代表格 Webpack といった多数のパッケージがあります。

npm の基本

npm の動作確認

npm は Node.js をインストールすると一緒にインストールされます。 npm のバージョンを確認してみましょう。

```
% npm -v
6.14.5
```

package.json とは

Node.js プロジェクトを管理するには、 package.json というファイルが必要です。 package.json は、JSON形式の設定ファイルで、 package.json を設定することで Node.js のパッケージをインストールをしたり、アプリのビルド、起動など様々な機能が利用できます。

npm の初期化

Node.js プロジェクト作成するには、プロジェクトフォルダ内で、 npm init コマンドを実行（初期化）します。

```
% mkdir プロジェクト名
% cd プロジェクト名
% npm init
```

npm init を実行すると、プロジェクト情報を設問形式で聞かれますが、今回はすべてエンターして進めていきます。 -y オプションで**設問をすべて YES**にすることもできます

```
...
package name: (プロジェクト名)
version: (1.0.0)
以降、設問に答える
...
```

package.json の確認

初期化が完了すると、**コマンドを実行したフォルダ内に package.json が作成**されるので確認してみましょう。

package.json の内容

```
{
  "name": "sample",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

npm パッケージのインストール

`npm` コマンドでパッケージをインストールします。**パッケージは node_modules フォルダにインストール**されます。

```
% npm install パッケージ名
//または
% npm i パッケージ名
```

package.json のデフォルトパッケージ

`package.json` にパッケージ情報を保存するには、以前は `--save` オプションを利用していましたが、 **ver5 以降は --save を省略**できるようになりました。`package.json` にはパッケージ情報 `dependencies` の欄に記載されます。

package.json の内容

```
{
  "name": "sample",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

package-lock.json とは

`npm` でインストールすると `package-lock.json` ファイルも作成されています。`package-lock.json` はパッケージのバージョンを固定するために利用され、**原則手動で編集してはいけません。** ファイルの中身を確認すると、本パッケージと依存パッケージのバージョン情報などが記載されています。

```
...
"express": {
```

```
  "version": "4.17.1",
...
```

一方 package.json のバージョンは範囲指定なので、実際のバージョンと異なるかも知れません。

```
...
"express": "^4.17.1"
...
```

本番用と開発用にわけてインストール

プロジェクトによっては本番では利用せずに開発だけパッケージを利用することもあり、本番用と開発用にわけてインストールすることができます。

開発用パッケージインストール

```
% npm i パッケージ名 --save-dev
//または
% npm i パッケージ名 -D
```

package.json には、 devDependencies 欄に記載されます。

本番用パッケージインストール

開発から本番に移動したい場合、 --production オプションをつけます。

```
% npm i --production パッケージ名
//または
% npm i -P パッケージ名
```

グローバルでパッケージをインストールする

毎回利用するパッケージをプロジェクトごとにインストールをするのが面倒な場合は、グローバルでインストールすることも可能です。

```
% npm i --global パッケージ名
//または
```

```
% npm i -g パッケージ名
```

ただし、グローバルインストールで設定すると、第三者にはわからないためプロジェクトを配布するときは注意が必要です。`npm` インストールを実行した時に、**パッケージがみつからないためエラー**になることがあります。

npm パッケージの確認と検索

依存するパッケージを含めたパッケージ一覧表示

通常パッケージ

```
% npm ls
```

グローバルパッケージ

```
% npm ls -g
```

トップレベルのパッケージ一覧表示

パッケージ一覧が多すぎてみずらいときは、トップレベルのパッケージを表示することもできます。

通常パッケージ

```
% npm ls --depth=0
```

グローバルパッケージ

```
% npm ls -g --depth=0
```

パッケージの検索

インストール可能なパッケージをキーワードで検索します。 ただ、これもみずらいので通常はネットで検索したりツールを利用することが多いでしょう。

```
% npm search キーワード
```

パッケージの詳細

パッケージ名を指定して、バージョン情報や依存パッケージなど表示できます。

```
% npm show パッケージ名
```

パッケージのアンインストール

パッケージをアンインストールは uninstall を利用します。

```
% npm uninstall パッケージ名
```

開発から削除

```
% npm uninstall パッケージ名 --save--dev
```