JavaScript応用実習

04. Node.jsとnpm

株式会社ジードライブ

© ZDRIVE, K.K. All rights reserved.

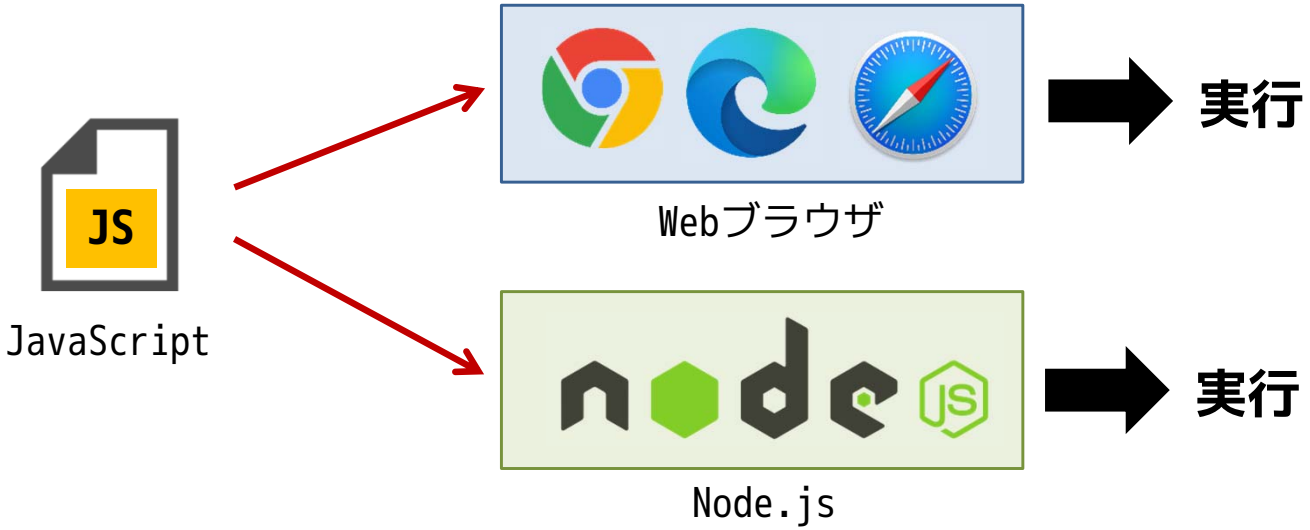
今回学ぶこと

- Node.jsについて
- npmとパッケージ管理

© ZDRIVE, K.K. All rights reserved.

JavaScriptの実行環境

- JavaScriptはブラウザやNode.jsで実行できる



  - ただし、実行環境により利用可能な操作が一部異なる

操作	ブラウザ	Node.js
ファイル操作	×	○
DOM操作	○	×
コンソール出力	○	○

© ZDRIVE, K.K. All rights reserved.

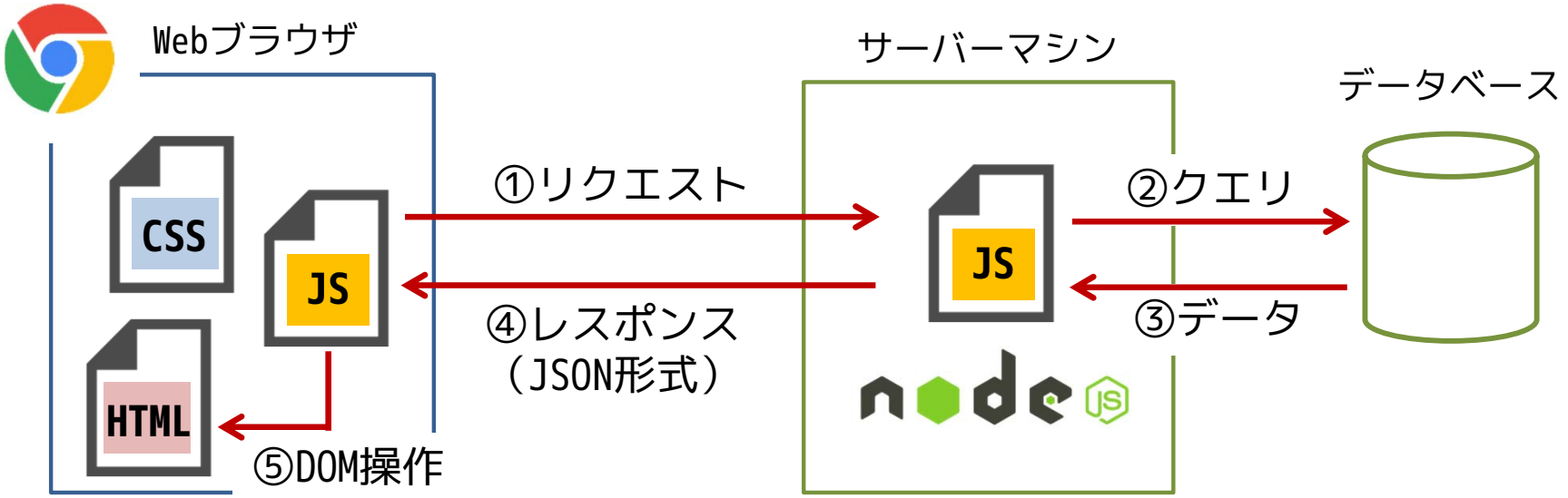
Node.jsについて

- Node.jsはサーバー用アプリケーション構築のために設計されたJavaScript実行環境
  - サーバー上でJavaScriptを動かすための環境
    ⇒ 代替の実行環境としては、DenoやBunが存在する
- 近年ではクライアント用アプリケーションの開発環境としても用いられている
  - 研修では、おもにこちらの使い方をする
- パッケージと呼ばれる様々なライブラリやプログラムを利用してアプリケーションの開発が行える

© ZDRIVE, K.K. All rights reserved.

Node.jsの利用場面①

- サーバーマシンにNode.jsをインストールし、データベースと連携するためのJavaScriptを実行する
  - 旧来、サーバー側のプログラムは、JavaやPHPといった言語で記述されていたが、Node.jsの登場により、サーバー側でもJavaScriptの利用が可能になった



© ZDRIVE, K.K. All rights reserved.

Node.jsの利用場面②

- 開発用マシンにNode.jsをインストールし、webpackやjson-serverなど開発を支援するツールを動かす
  - webpackはモジュールバンドラと呼ばれるツールで、複数のJavaScript等のファイルをひとまとめにすることができる
  - json-serverは簡易的なAPIを作成するためのツール

フロントエンド(ブラウザで実行するJavaScript)の開発であっても、Node.jsは欠かすことができない存在になっている

© ZDRIVE, K.K. All rights reserved.

Node.jsとnpm

- npmは、JavaScriptの実行環境であるNode.js関連のコマンドラインツールで、パッケージ(ライブラリ)を管理する際に使用する
  - **N**ode **P**ackage **M**anager
- npmは、package.jsonという設定ファイルに従って、パッケージを管理し、プロジェクト内で利用できるようにする
  - NPMリポジトリから必要なパッケージをダウンロードする
  - あるパッケージが別のパッケージに依存している場合、その依存関係を解決する(この情報は、package-lock.jsonに記載される)

npmのインストール確認(コマンドプロンプトに黄字部分を入力し、Enter)

```
C:¥Users¥zd1A04> npm -v
10.2.4
```

© ZDRIVE, K.K. All rights reserved.

npmコマンドの実行

- npmコマンドを使用する場合は、VS Code上のターミナルで実行する便利
  - VS Codeで開いているプロジェクト内に必要なファイルがダウンロードされる

© ZDRIVE, K.K. All rights reserved.

package.jsonの生成

- npmをプロジェクト内で利用するには、以下のコマンドを実行する
  - このコマンドによって、プロジェクトの情報を記したpackage.jsonが生成される

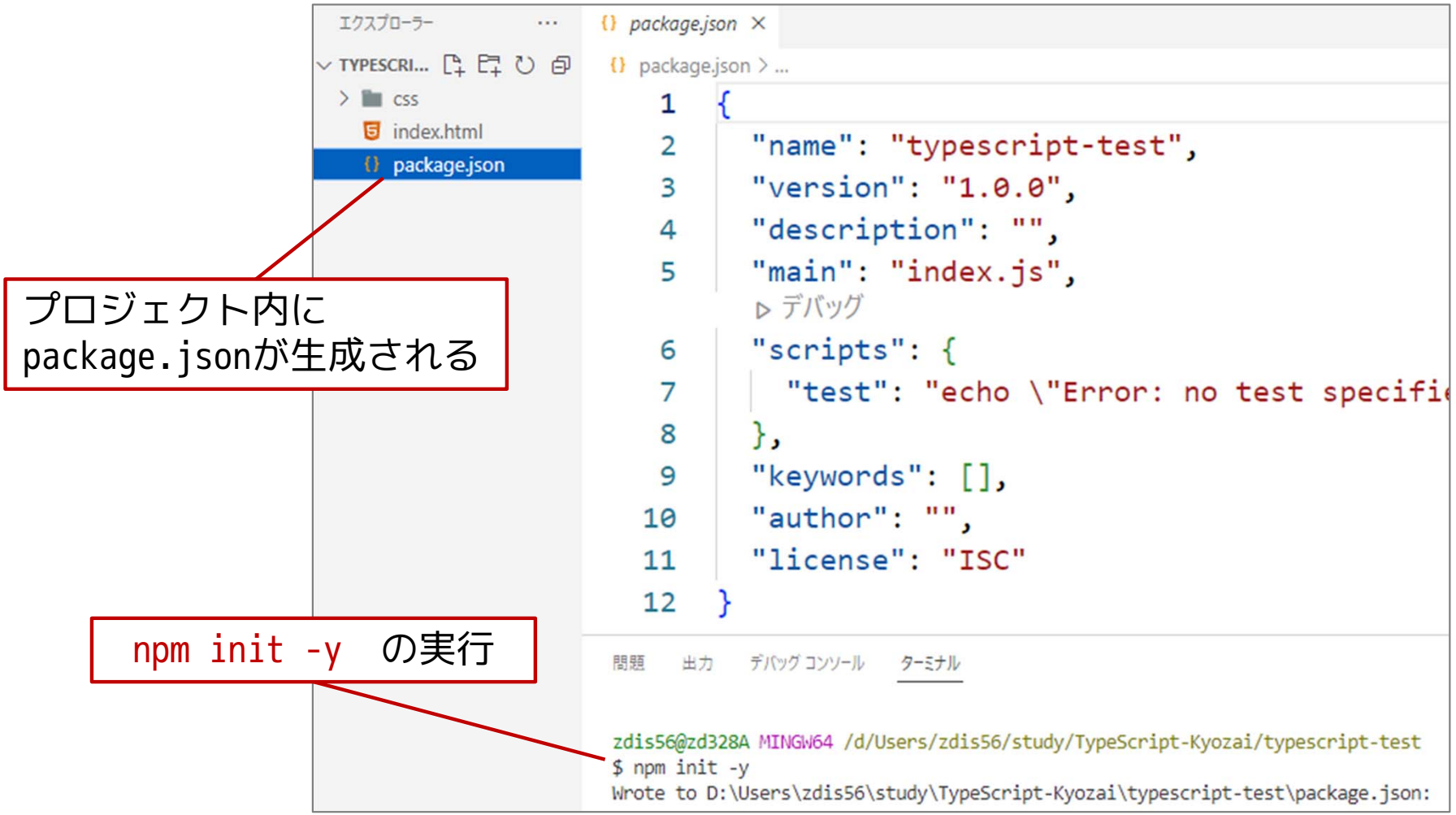
```
npm init -y
```

  - VS Codeで複数のプロジェクトを開いている場合は、cdコマンドを使い、あらかじめプロジェクトのルートとなるフォルダをカレントディレクトリにしておく
  - -y オプションを付けない場合、一つずつ質問に答えながら、package.jsonファイルを作成していくことになる

© ZDRIVE, K.K. All rights reserved.

package.jsonの生成

- 生成されるpackage.jsonの例



© ZDRIVE, K.K. All rights reserved.

package.json

- package.json には、JSON形式で以下のような情報が記述される

プロパティ	説明
name	プロジェクト名
version	プロジェクトのバージョン
description	プロジェクトの説明
main	プロジェクトのエントリーポイントとなるファイル
scripts	`npm run ○○` で実行されるコマンドを記述する
dependencies	本番環境で必要なパッケージを記載する
devDependencies	開発時に利用するパッケージを記載する

© ZDRIVE, K.K. All rights reserved.

パッケージのインストール

- システムの開発に際して必要なパッケージは、`npm install` コマンドを使ってインストールできる
- パッケージには、以下のような種類が存在する
  - システムの**動作**に必要な機能を提供するパッケージ
  - システムの**開発**に必要な機能を提供するパッケージ
- インストールの方法は、以下の2種類が存在する
  - **ローカルインストール**：
    プロジェクト内でのみ使用可能なパッケージをインストールする
  - **グローバルインストール**：
    全てのプロジェクトで使用可能なパッケージをインストールする

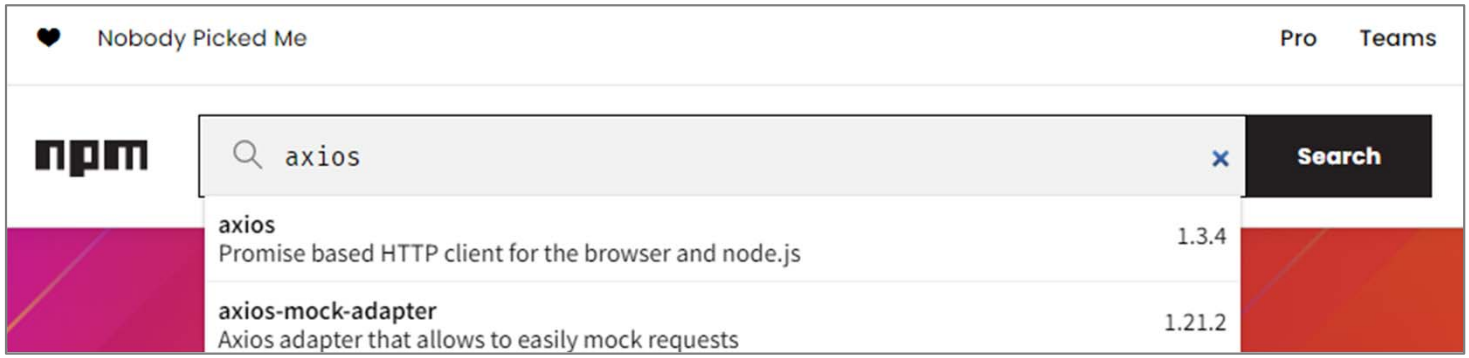
© ZDRIVE, K.K. All rights reserved.

パッケージのインストール

- プロジェクト内にパッケージをインストールするには、以下のコマンドを実行する
  - `install` は、省略して `i` と記述してもよい
  - 特定のバージョンを指定する場合、@バージョンが必要

```
npm install パッケージ名@バージョン
```

- パッケージ名やバージョン、パッケージの使い方等は、https://www.npmjs.com/ で調べることができる



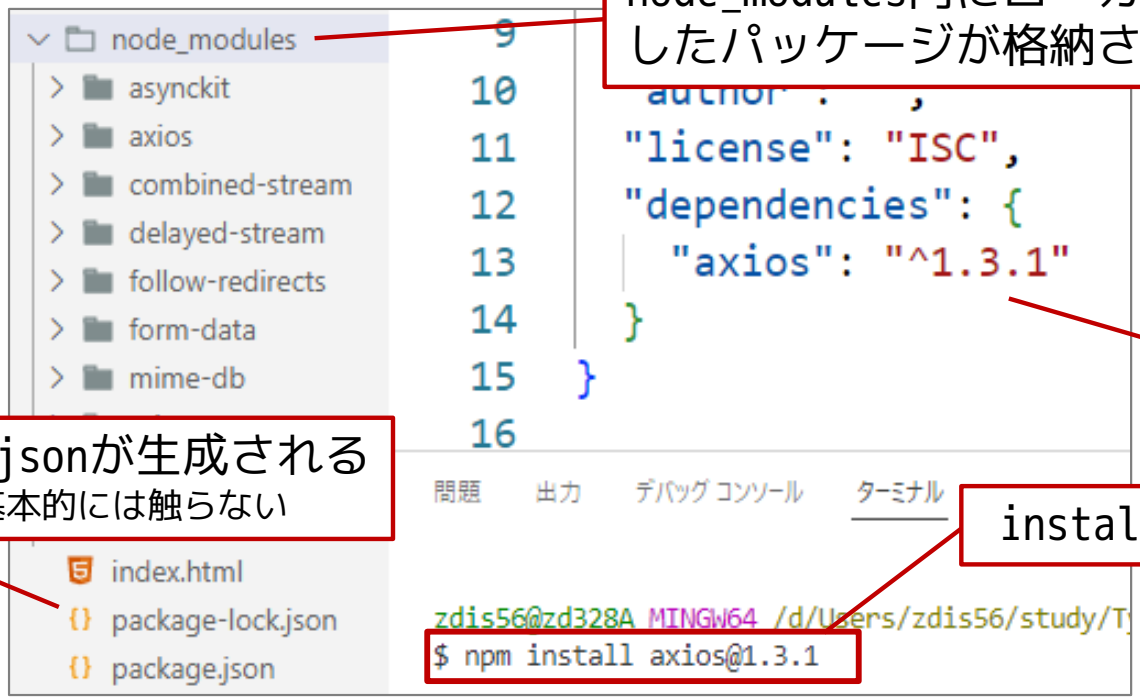
© ZDRIVE, K.K. All rights reserved.

パッケージのインストール

- installコマンドを実行すると、
  ① package.jsonに追記が行われる
  ② package-lock.jsonが生成される
  ③ node_modulesフォルダが生成され、パッケージがインストールされる

package.jsonがない場合は、生成される

node_modules内にローカルインストールしたパッケージが格納される



package.jsonに記述が追加される

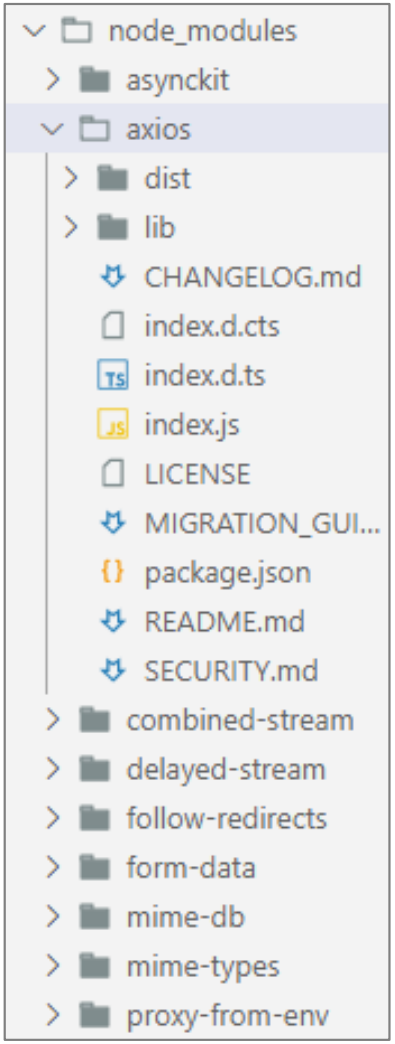
package-lock.jsonが生成される
このファイルは、基本的には触らない

installコマンドを実行

© ZDRIVE, K.K. All rights reserved.

node_modules

- node_modulesには膨大な量のフォルダ／ファイルが含まれる
  - これらのファイル群は、Gitでバージョン管理をしない
    ⇒ Github上にプッシュしない
    ⇒ FileZilla等で、家に持って帰らない



- プロジェクト内にpackage.jsonがあれば、installコマンドで、node_modulesフォルダが生成され、パッケージのダウンロードも行われる

既存のpackage.jsonに基づき、パッケージをインストール

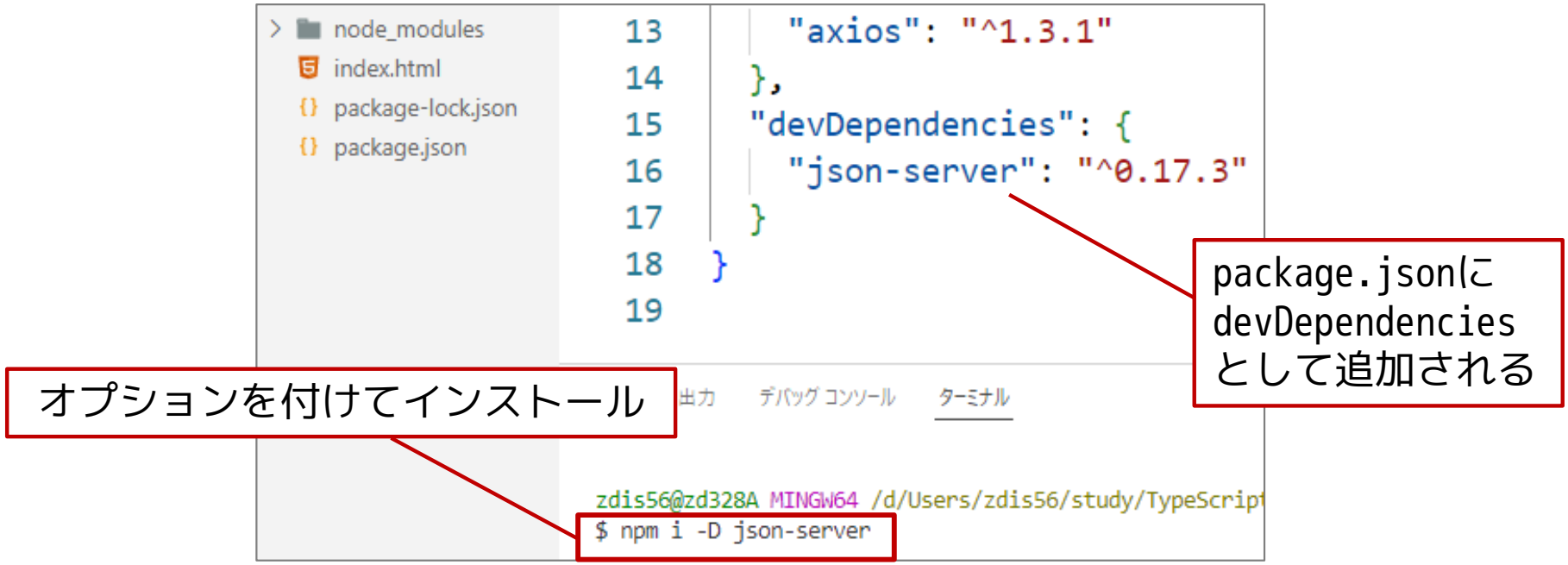
```
npm install
```

© ZDRIVE, K.K. All rights reserved.

--save-devオプション

- 開発時にのみ必要なパッケージは、--save-devオプションを付けてインストールする
  - オプションは省略して、**-D** と記述することもできる

```
npm install --save-dev パッケージ名@バージョン
```



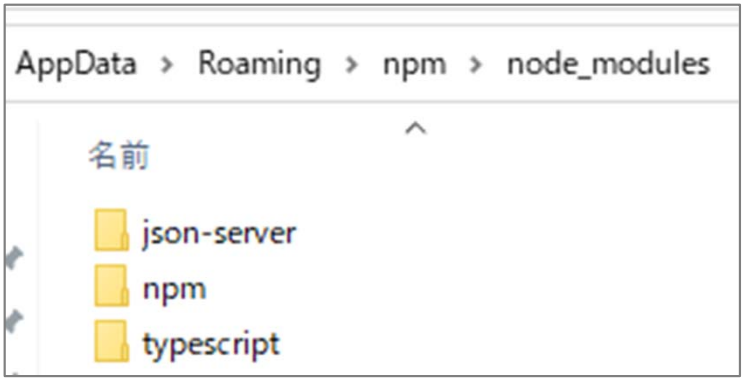
© ZDRIVE, K.K. All rights reserved.

-gオプション

- パッケージをグローバルインストールするには、**-g** オプションを付ける
  - このオプションが無い場合、ローカルインストールになる

```
npm install -g パッケージ名@バージョン
```

- -g オプションを付けると、プロジェクトルートではなく、npm 配下のルートフォルダにインストールされる
  - Windowsの場合、自身のユーザーフォルダのAppData内にダウンロードされる



© ZDRIVE, K.K. All rights reserved.

オプションの使い分け

オプション	説明
`-g`	特定のプロジェクトに限定されない開発補助ツールのインストール時に利用する。 例: `create-react-app, nodemon, json-server` ※ グローバルインストールしたツールについては、`package.json`に依存関係が記述されない。つまり、他の端末や開発者と共有されないという点に注意する
`--save-dev (-D)`	開発を補助するツールのインストール時に利用する。 例: `Jest, webpack, ESLint`
なし	アプリケーションそのものが必要としているライブラリやフレームワークのインストール時に利用する。 例: `Axios, React, Express`

© ZDRIVE, K.K. All rights reserved.

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する場合は、以下のコマンドを実行する

```
npm uninstall パッケージ名
```

- -gオプションを付けてインストールしたパッケージは、オプションを付けてuninstallコマンドを実行する

```
npm uninstall -g パッケージ名
```

© ZDRIVE, K.K. All rights reserved.

npxコマンド

- ローカルインストールしたパッケージに含まれるコマンドを実行する
  - パッケージがインストールされていない場合は、自動的にインストールが行われ、コマンド実行後に削除される

```
npx コマンド
```

「cowsay」の実行例

```
npx cowsay Hello
Ok to proceed? (y) y
< Hello >
 -------
        ¥   ^__^
         ¥  (oo)¥_______
            (__)¥       )¥/¥
                ||----w |
                ||     ||
```

© ZDRIVE, K.K. All rights reserved.

npxコマンド

- 前頁のコマンド実行時に以下のようなメッセージが表示される場合、手動でnpmフォルダを作成する必要がある
  - 自身のユーザフォルダ内のAppData/Roaming内に作成する

```
npm ERR! enoent ENOENT: no such file or directory, lstat
`C:¥users¥自身のユーザフォルダ¥AppData¥Roaming¥npm`
```

© ZDRIVE, K.K. All rights reserved.

グローバルインストールと実行コマンド

- グローバルインストールしたものについては、実行時にnpxは必要ない

「cowsay」のグローバルインストールと実行例

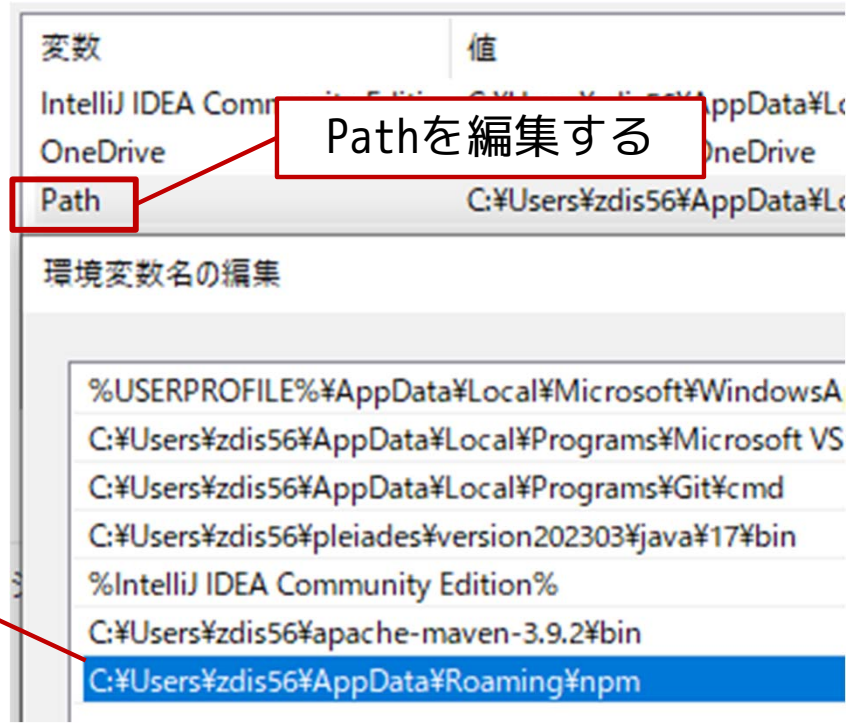
npxは
書かない

```
npm install -g cowsay

cowsay Hello

< Hello >
 -------
        ¥   ^__^
         ¥  (oo)¥_______
            (__)¥       )¥/¥
                ||----w |
                ||     ||
```

© ZDRIVE, K.K. All rights reserved.

グローバルインストールと実行コマンド

- グローバルインストールしたものが実行できない場合は、環境変数を設定する
  - 設定後は、VS Codeを再起動
  - VS Codeを再起動しても問題が解決しない場合は、VS Codeを再インストールする



C:¥Users¥自身のユーザフォルダ¥AppData¥Roaming¥npm

© ZDRIVE, K.K. All rights reserved.

scriptsプロパティ

- package.json内のscriptsプロパティには `npm run ○○` で実行したいコマンドを記述する

```
"main": "index.js",
▷ デバッグ
"scripts": {
    "start": "npx cowsay Start!",
    "dev": "npx cowsay Develop!",
    "test": "npx cowsay Test!"
},
"keywords": []
```

npm start
で実行できる
※ **run** は省略可

npm run dev
で実行できる

npm test
で実行できる
※ **run** は省略可

実行したコマンドをキャンセル(停止)するには、
ターミナル上で **Ctrl + C** を入力する

© ZDRIVE, K.K. All rights reserved.

練習

- 練習04-1

© ZDRIVE, K.K. All rights reserved.

補足：その他のnpmのコマンド

© ZDRIVE, K.K. All rights reserved.

パッケージの検索・情報の表示

- npmリポジトリからパッケージを検索する

```
npm search パッケージ名
```

- パッケージに関する各種情報を表示する

```
npm view パッケージ名
```

- パッケージのリリースバージョンの一覧を表示する

```
npm view パッケージ名 versions
```

© ZDRIVE, K.K. All rights reserved.

インストール済みパッケージの確認

- パッケージの一覧を表示する

```
npm list
```

```
npm list -g
```

  - パッケージ名とバージョンが表示される
  - **-g**を付けるとグローバルインストールしたパッケージを対象とし、**-g**を付けない場合はローカルインストールしたパッケージを対象とする
  - **list** は省略して、**ls**と記述することもできる

© ZDRIVE, K.K. All rights reserved.

インストール済みパッケージの確認

- インストール済みパッケージの一覧を表示する
  - パッケージ名、バージョン、簡単な説明
  - `la` は `ll` でもよい

```
npm la
```

```
npm la -g
```

- 直接インストールしたパッケージのみ表示する

```
npm list --depth=0
```

```
npm list -g --depth=0
```

© ZDRIVE, K.K. All rights reserved.

パッケージの更新

- バージョンが古くなった未更新のパッケージを確認する

```
npm outdated
```

```
npm outdated -g
```

- package.jsonに記載されているパッケージのバージョンに更新する

```
npm update
```

© ZDRIVE, K.K. All rights reserved.

パッケージの更新

- package.jsonのパッケージのバージョンを一括で最新にする
  - 準備として、npm-check-updatesをインストールしておく

```
npm install -g npm-check-updates
```

- package.jsonを一括更新する
  - npm-check-updatesは「ncu」と省略してもよい

```
npm-check-updates -u
```

  - その後、以下のコマンドを実行する

```
npm update
```

© ZDRIVE, K.K. All rights reserved.