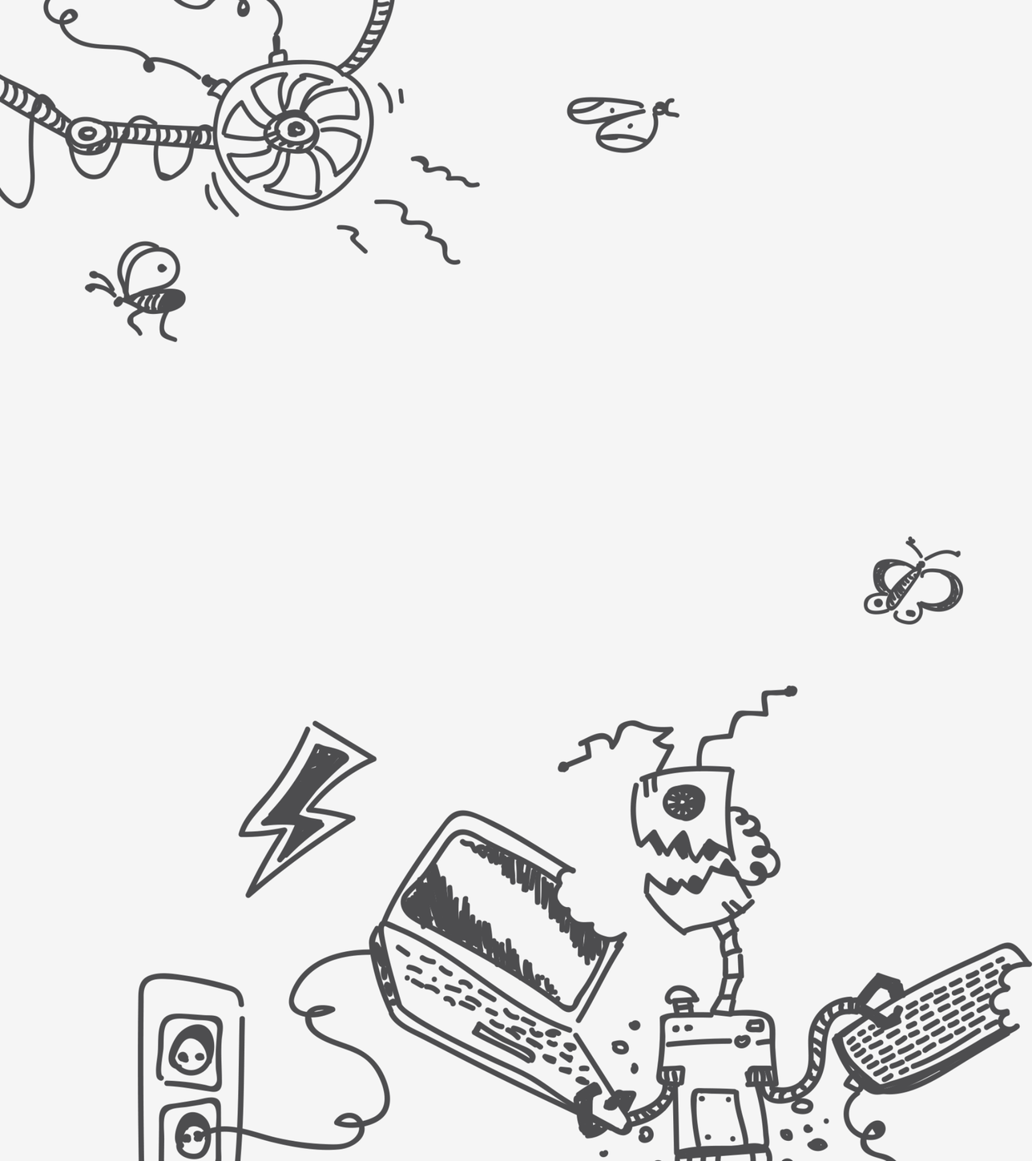


Zarządzanie zależnościami

Zarządzanie zależnościami

Zarządzanie zależnościami jest kolejną cechą menedżerów pakietów.

Przy większym projekcie będziemy korzystać z wielu pakietów, innych dla wersji produkcyjnej naszego projektu, innej dla wersji testowej, a jeszcze innej dla wersji deweloperskiej.

Nieefektywne byłoby ręczne instalowanie każdego pakietu za każdym razem, kiedy nowa osoba trafia do projektu albo kiedy projekt jest migrowany na inny serwer.

Plik package.json

Npm rozwiązuje ten problem dzięki plikowi o nazwie `package.json` (plik musi się tak nazywać).

Plik ten zbiera wszystkie informacje na temat naszego projektu. W nim wpisane będą też wszystkie paczki potrzebne do działania.

Poprawne przygotowanie i korzystanie z pliku `package.json` pozwoli szybko zainstalować wszystkie paczki, i to po wpisaniu tylko jednej komendy.

Inicjalizacja pliku package.json

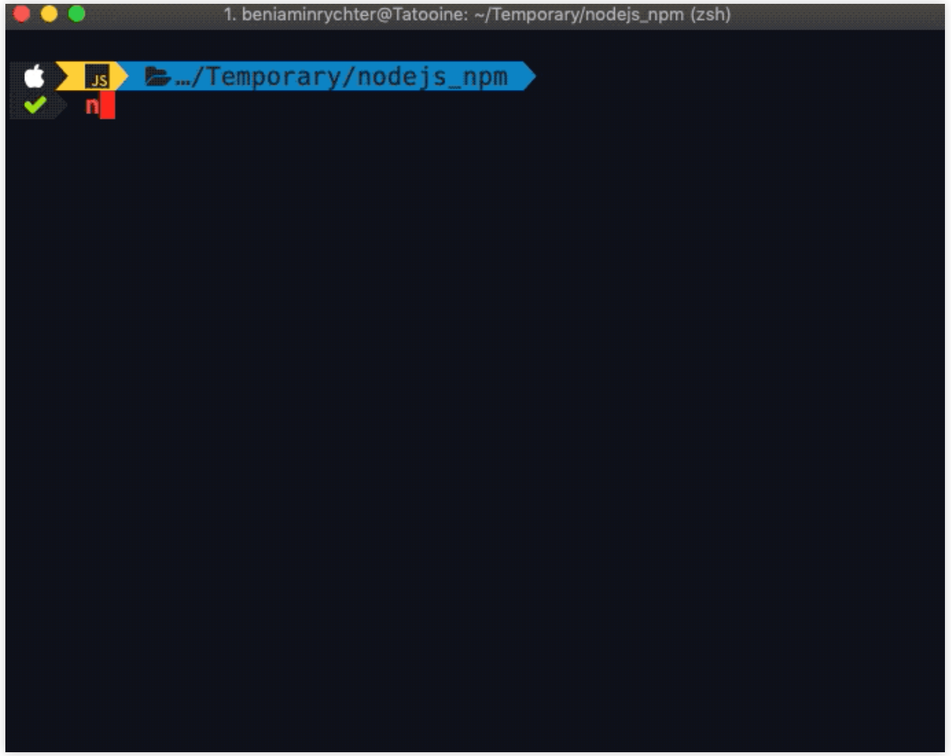
Plik `package.json` możemy napisać ręcznie sami albo skorzystać z generatora, który go dla nas stworzy. Aby uruchomić generator należy użyć komendy:

```
npm init
```

Komenda ta musi zostać wywołana w głównym katalogu naszego projektu.

Po uruchomieniu tej komendy będziemy musieli podać jej wszystkie potrzebne informacje. Odbywa się to w interaktywny sposób.

Pod koniec zostanie nam wyświetlone podsumowanie wygenerowanego pliku.



Najważniejsze pola w pliku package.json

```
{
  "name": "nodejs",
  "version": "1.0.3",
  "description": "Jakiś opis aplikacji",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Test\""
  },
  "author": "Coders Lab",
  "license": "ISC",
  "dependencies": {
    "foo" : "1.0.0",
    "nazwa_paczki" : "wersja_paczki"
  },
  "devDependencies": {
    "bar" : "1.0.0"
  }
}
```

- `name` - Nazwa naszej paczki. Pole obowiązkowe, które musi składać się z liter (małych) i ew. podkreśleń.
- `version` - Wersja naszej paczki. Pole obowiązkowe, które powinno składać się z trzech liczb całkowitych oddzielonych kropkami.
- `scripts` - Obiekt zawierający komendy możliwe do uruchomienia poprzez `npm`.
- `dependencies` - Nazwy paczek (w postaci tablicy JSON), które są potrzebne do działania dla wersji produkcyjnej naszego projektu (kalendarze, wykresy, itp.).
- `devDependencies` - Dodatkowe paczki (w postaci tablicy JSON), które są potrzebne przy tworzeniu naszego projektu (np. ESLint, TypeScript itp.).

Zależności i ich wersje

Zależności w pliku `package.json` wpisujemy w następujący sposób:

```
{
  "dependencies" : {
    "foo" : "1.0.0",
    "nazwa_paczki" : "wersja_paczki"
  }
}
```

Zależności i ich wersje

Istnieje wiele sposobów podawania wersji paczki. Najpopularniejsze to:

Dokładna wersja

Musimy podać dokładną wersję w formacie `"X.Y.Z"`.

Najnowsza wersja z rodziny

Możemy pominąć któryś z numerów wersji i wstawić tam gwiazdkę. Dzięki temu `npm` ściągnie najnowszą wersję z tej rodziny.

Wersja co najmniej

Jeżeli przed numerem wersji dodamy znak `>=`, to zostanie ściągnięta co najmniej podana wersja (ale może być też każda nowsza – domyślnie zainstaluje najnowszą).

Wersja najwyżej

Jeżeli przed numerem wersji dodamy znak `<=`, to zostanie ściągnięta co najwyżej podana wersja (ale może być też każda starsza).

Zależności i ich wersje

Zależności w pliku `package.json` wpisujemy w następujący sposób:

```
{
  "dependencies" : {
    "foo" : "1.0.0", <- paczka o nazwie foo w wersji 1.0.0
    "bar" : ">= 2.3.5", <- paczka o nazwie bar w wersji co najmniej 2.3.5
    "baz" : "4.*", <- paczka o nazwie baz w najnowszej wersji z rodziny 4
    "baz2" : "1.2.*", <- paczka o nazwie baz2 w najnowszej wersji z rodziny 1.2
    "foobar" : "*", <- paczka o nazwie foobar w najnowszej wersji (ogólnie)
  }
}
```

Więcej o wersjonowaniu semantycznym (bo tak nazywa się ten sposób wersjonowania) możesz przeczytać tutaj:

→ http://semver.org

→ http://docs.npmjs.com/misc/semver

Dodawanie zależności do package.json

Do pliku `package.json` możemy dodawać zależności ręcznie, po prostu je wpisać. Możemy też skorzystać z poznanej wcześniej komendy konsolowej `i`:

```
npm i <nazwa paczki>
```

lub:

```
npm install <nazwa paczki> --save-dev
```

Komenda `i` automatycznie dołączy naszą paczkę do pola `dependencies` w naszym pliku.

Flaga `--save-dev` dołączy naszą paczkę do pola `devDependencies` w naszym pliku.

Instalacja wszystkich zależności

Aby zainstalować wszystkie zależności w przypadku, w którym mamy istniejący (i wypełniony) plik `package.json`, musimy użyć komendy:

```
npm i
```

Zostaną automatycznie doinstalowane wszystkie brakujące zależności.