Co nowego w npm?

by Łukasz Rybka

Agenda

- npm install --save
- package-lock.json
- npm ci
- offline/online mode
- npx
- npm init
- npm audit

npm install --save (1)

- Aby zapisać instalowaną zależność w package.json do tej pory wykonywaliśmy polecenie:

npm install --save lodash

- Teraz flaga --save jest domyślna, a więc powyższe polecenie można skrócić:

npm install lodash

npm install --save (2)

- Aby nie zapisywać zależności należy użyć flagi --no-save

npm install --no-save lodash

npm install --save (3)

- Informacje czy zależność jest automatycznie zapisywana można sprawdzić w konfiguracji npm:

npm config get save

- Aby wyłączyć automatyczne zapisywanie zależności wystarczy zmiana jednego ustawienia w konfiguracji npm:

npm config set save false

npm install - SemVer

- npm używa SemVer do określenia wersji zależności
- Format: **MAJOR**.**MINOR**.**PATCH**
  - wersję **MAJOR**, gdy dokonujesz zmian niekompatybilnych z API,
  - wersję **MINOR**, gdy dodajesz nową funkcjonalność, która jest kompatybilna z poprzednimi wersjami,
  - wersję **PATCH**, gdy naprawiasz błąd nie zrywając kompatybilności z poprzednimi wersjami.

npm install - typowe problemy

- Małe zrozumienie faktycznej wersji instalowanej
- Brak zrozumienia mechanizmu Shrinkwrap
- Liczne problemy z Shrinkwrap
- Długi czas instalacji (poświęcany na analizę drzewa zależności)

npm install - package-lock.json

- Domyślnie generowany w npm 5.X.X
- Zawiera informacje o dokładnych wersjach zależności oraz ich zależności
- Aktualizacja zależności wymaga ponownej instalacji zależności
- **Musi być wersjonowana razem z plikiem package.json!**
- Wymagane do przeprowadzania audytów bezpieczeństwa!
- Generowany (aktualizowany) po każdej operacji zmieniającej zawartość katalogu node_modules lub pliku package.json

npm ci

- Służy do instalowania zależności projektu “na czysto” w możliwie szybki sposób
- Wymaga pliku package-lock.json
- Komenda stworzona z przeznaczeniem dla środowisk takich jak CI, CD
- Komenda bardziej restrykcyjna od npm install

npm install vs npm ci

- npm install nie wymaga pliku lock, ci tak
- Jeżeli w trakcie instalacji zależności npm ci zauważy różnicę między plikiem lock a package.json - operacja zostanie przerwana oraz błąd będzie zwrócony
- npm ci instaluje zależności całego projektu, instalacja pojedynczej paczki nie jest możliwa
- Jeżeli katalog node_modules istnieje lokalnie przed wykonaniem komendy npm ci - zostanie on usunięty i utworzony na nowo
- npm ci nie modyfikuje plików lock oraz package.json

Offline/online modes

- **--prefer-offline** - opcja sprawiająca, że dla każdego zasobu znajdującego się w (nowym) cache'u połączenie sieciowe nie zostanie wykonane
- **--prefer-online** - opcja która zmusza npm do ponownej walidacji cache'a i zaktualizowania go nowościągniętymi paczkami
- **--offline** - opcja pobierająca zależności wyłacznie z lokalnego cache'u. W przypadku brakującej lokalnie paczki błąd z kodem ENOTCACHED zostanie zwrócony a instalacja przerwana

npx (1)

- Narzędzie automatycznie instalowane z npm w wersji 5.2.0 lub wyższej
- Służy do uruchamiania skryptów paczek npm'owych
- Automatycznie pobiera wszystkie zależności paczki której skrypt chcemy uruchomić (dawniej musieliśmy instalować paczkę globalnie/lokalnie)
- Wykorzystywana paczka nie jest instalowana globalnie, przestrzeń nazw jest znacznie czystsza
- Upraszcza uruchamianie lokalnych skryptów

npx (2)

- Wykorzystywana paczka nie jest instalowana globalnie, przestrzeń nazw jest znacznie czystsza

npx workin-hard

npx (3)

- Upraszcza dzielenie się skryptami (np. GitHub Gist):

npx https://gist.github.com/zkat/4bc19503fe9e9309e2bfaa2c58074d32

npx (4)

- Pozwala na uruchamianie komend z różnymi wersjami Node

npx -p node@7 npm run build

npm init

- Od wersji 6.1.0 npm init posiada nową opcję - <initializer>

npm init react-app playground

- npm wyszuka w repozytorium paczki o nazwie wskazanej jako inicjator (react-app) z prefixem create (konwencja)
- paczka zostanie pobrana za pomocą narzędzia npx i odpowiedni skrypt binarny zostanie uruchomiony (również za pomocą npx)

npm audit

- Od wersji 6 npm posiada wbudowany mechanizm weryfikacji bezpieczeństwa naszych zależności - npm audit
- Każda paczka jest analizowana pod względem zagrożeń bezpieczeństwa w bazie Node Security Platform (autorstwa ^Lift Security, obecnie część npm, Inc.)
- Dla każdej wrażliwości (niezależnie od jej stopnia) otrzymujemy informacje o źródle, bazowej zależności, informacji w bazie Node Security i sposobie naprawy (jeśli istnieje)
- **Szczególnie ważne w kontekście ostatniej afery z flatmap-stream!**

Zasoby

- Blog npm - wartości wynikające z używania SemVer w projektach: https://blog.npmjs.org/post/162134793605/why-use-semver
- Dokumentacja npm - wyjaśnienia działania i potrzeby lockfiles: https://docs.npmjs.com/files/package-locks

Pytania?

Dzięki!

https://infoshareacademy.com

http://www.dragonia.org.pl