TypeScript, Node.js Project Setup, Introduction to Microservices

Node.js Accelerator – Jan'23

Agenda

1. NPM packages review
2. NPM: Versioning
3. Project Setup
4. Introduction to microservice
5. Debugging
6. Useful Packages

Toptal®

Node.js Accelerator

Jan’2023

- dependencies: Packages required by your application in production.
    - npm i express
    - npm i –save express (previous npm versions)
    - yarn add express
- devDependencies: Packages that are only needed for local development and testing.
    - npm i –save-dev nodemon
    - yarn add -D nodemon
- Global dependency: package that can be installed locally once and used in different projects:
    - npm i -g nodemon
    - yarn global add nodemon

Dependencies versioning are followed in x.y.z format. A initial release usually goes for 1.0.0 where it is defined by

- Major release (**1**.0.0): when 1 is for the new product and each increment represents a major change in the dependency. Once a new major release is set, the patch and minor releases should go back to 0.

  Example: from 1.3.7 to 2.0.0

- Minor release (1.**0**.0): this represents any new features from a new released version of the package that doesn't contain breaking changes from previous features. When released, the last digit should go back to 0.

  Example: from 1.0.19 to 1.1.0

- Patch release (1.0.**0**): related to any backward compatibility bug fixes that are released.

  Example: from 1.0.0 to 1.0.1

NPM: Versioning

To control what we can update or not in npm as dependencies:

- Patch releases: 1.0 or 1.0.x or ~1.0.4
- Minor releases: 1 or 1.x or ^1.0.4
- Major releases: * or x

Patch and minor releases are usually fine to be updated.

Watch out for major releases that can break your app!

```
"devDependencies": {
 "@types/node": "^18.7.18",
 "ts-node": "~10.9.1",
 "typescript": "x"
},
```

NPM: Listing

- You can verify your dependencies as well as dependencies of your dependencies with:
    - npm list: your dependencies
    - npm list --depth=n: your dependencies and going in depth (switch **n** to a desirable number)

```
~/projects/samples-js ❱ npm list
Node 16.17.0 15:56:35
samples@1.0.0
/Users/marcossilva/projects/samples-js
├── @types/node@18.7.18
├── slugify@1.6.5
├── ts-node@10.9.1
├── typescript@4.8.3
└── url@0.11.0
```

NPM: Updating

You can verify outdated dependencies with **npm outdated** command

```
~/projects/samples-js ❱ npm outdated
Node 16.17.0 16:14:17
Package       Current    Wanted   Latest  Location                 Depended by
@types/node  16.11.59  16.11.59  18.7.18  node_modules/@types/node  samples-js
```

You can verify outdated dependencies with **npm update** command

```
~/projects/samples-js ❱ npm update
changed 1 package, and audited 24 packages in 723ms
found 0 vulnerabilities
```

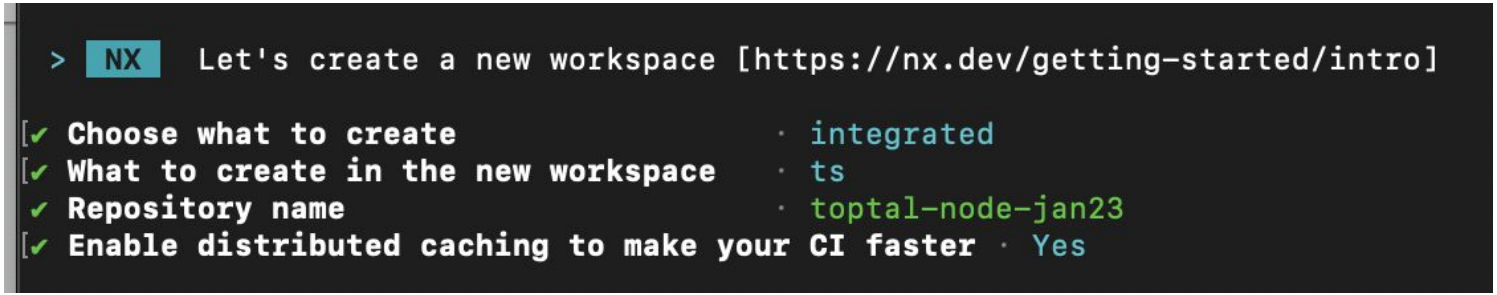
Publishing receipt:

- Create your node.js project
- Configure your git repository
- Define the name of your project
- Configure the build
- Login to npm
- Publish your project
    - npm publish
    - [optional] Use np library to facilitate the process

Let’s setup a project with monorepo approach using nx.dev

Creating a workspace:

- npx create-nx-workspace



- npm install -D @nrwl/node

More on monorepo

Installing your app:

- nx g @nrwl/node:application pure-ts-app
- Serving your app: nx serve pure-ts-app

```
> nx run pure-ts-app:serve

Debugger listening on ws://localhost:9229/db59685c-2479-48c9-afc9-44010fd09dec
Debugger listening on ws://localhost:9229/db59685c-2479-48c9-afc9-44010fd09dec
For help, see: https://nodejs.org/en/docs/inspector
Hello World
```

More on monorepo

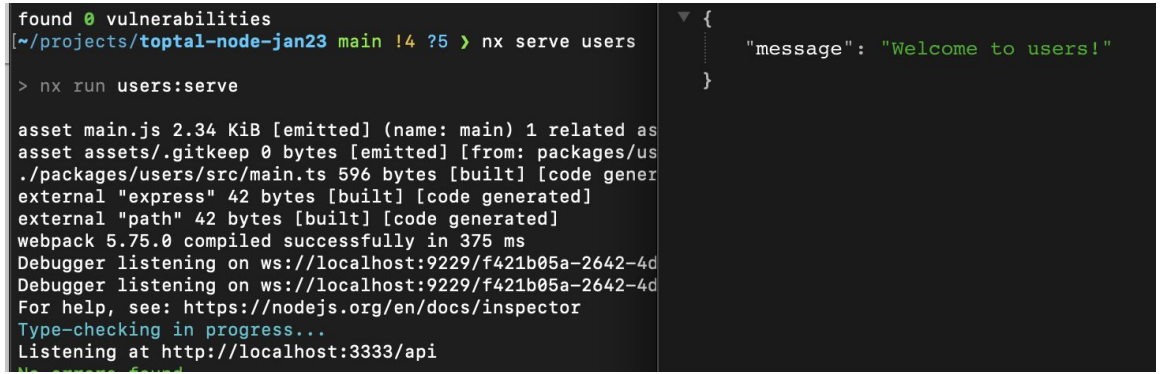
Installing a express application in your monorepo:

- npm i --save-dev @nrwl/express

- nx g @nrwl/express:app users

Testing it with:

- nx serve users



More on monorepo

Show debug methods using Webstorm / VSCode

- VSCode: https://code.visualstudio.com/docs/typescript/typescript-debugging
- Webstorm: https://www.jetbrains.com/help/webstorm/running-and-debugging-typescript.html

Here is some basic and useful packages for your daily work:

- nodemon: automatic restart your node application when code changes
- dotenv: loads environment variables from .env file
- chance: helps to mock data for your tests
- commitizen: git commit with conventions.
- prettier: an opinionated code formatter.
- np: a better npm publish
- @eslint/config: analyses your code and help you find issues!
- ts-node: TypeScript execution environment and REPL for node.js

Toptal®

ACTION ITEMS

Week 1

1. We strongly recommend you to explore below module through Udemy this week and learn about Node.js essentials

Type	Topic	Description	Udemy Link	Duration
Self-paced learning on Udemy	Introduction to Node.js and NPM	You will learn what Node.js is and what makes Node.js so popular. You will also learn how to use Node Package Manager (NPM) and Nodemon, installing node and creating your first app	Click Here	40 min
Self-paced learning on Udemy	Optional: Javascript refresher	A quick refresher on JavaScript, understand arrow functions, objects, properties, methods, a few ES6 features, async code & promises	Click Here	51 min
Self-paced learning on Udemy	Node.js Basics	Understand essential concepts in node.js, the node lifecycle & event loop, node.js processes, event-driven code execution, blocking & non-blocking code, node.js core modules, debugging	Click Here	2hr 35min


Toptal®