TS

Fully Revised for 2026

# Creating NPM Package

## Simplified TypeScript Guide to Building and Publishing Libraries

OLUWATOBI SOFELA

# Creating NPM Package with TypeScript

## Simplified TypeScript Guide to Building and Publishing Libraries

This book is available at
https://leanpub.com/creating-npm-package-with-typescript




First published in 2025

Revised and updated edition published in 2026

www.codesweetly.com

# Tweet This Book!

Please help CodeSweetly by spreading the word about this book on Twitter!

The suggested hashtag for this book is #typescript.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#typescript

## Also By CodeSweetly

Learn Coding Visually
Code React Sweetly
Creating NPM Package with Vanilla JavaScript
Creating NPM Package with React TypeScript
The CSS Grid Guidebook
Visual CSS Grid
Visual CSS Flexbox
Creating NPM Package with ReactJS
CSS Flexbox

# Contents

# Introduction

## Welcome to Creating NPM Package with TypeScript!

If you've ever installed a JavaScript library using `npm install`, you've used a package from the NPM registry (a global database where developers publish and share reusable code).

This book is your step-by-step guide to joining them.

No matter what you call it—a package, library, plugin, framework, or tool—an NPM package is basically a project folder with a `package.json` file. This file explains what the package does, what it needs, and how others can use it.

Publishing your own NPM packages is more than a technical milestone. It allows you to:

- Share your work with the world
- Contribute to the open-source community
- Showcase your ability to ship production-ready tools
- Strengthen your skills in areas like:
  - Communication
  - Collaboration
  - Organization
  - Documentation
  - Problem-solving

By the end of this book, you'll be able to build, test, and publish a TypeScript-based NPM package with confidence, using modern tools and workflows.

## Why This Book Will Help You

The most effective way to master package publishing is to build as you learn, rather than only reading about it.

That's why this book is hands-on and project-based. You'll write code as you learn each concept. By the end, you'll have a working, polished NPM package and the real-world experience to create more.

So open your code editor, such as VS Code, follow along, and build with me. This book is more than a tutorial. It's a launchpad.

Let's break free from tutorial hell and start shipping real things.

## What You'll Build

In this book, you'll build a Tweet Button package. It's a simple but scalable project that covers all the key parts of package development.

If the project sounds niche, don't worry. The techniques you'll learn apply to any kind of NPM package. Consider this project your sandbox. Feel free to build on it or branch out into your own ideas.

## Use the Right Tools

To avoid unexpected issues, it is advisable to use the exact versions of dependencies specified in this book. This approach allows you to focus on learning rather than troubleshooting problems caused by newer releases.

Once you've completed the book, feel free to upgrade and experiment. And if something breaks? That's your next learning opportunity.

## What You Should Know First

You will benefit most from this book if you are already comfortable with:

- JavaScript fundamentals
- Basic TypeScript syntax
- Git and GitHub basics

If these tools sound completely new, consider reviewing them first. But if you've ever committed code or written a `.ts` file, you're ready.

## Got Questions?

If you get stuck, have feedback, or just want to say hi, feel free to reach out! You can email me at contact@codesweetly.com or message me on Twitter (@oluwatobiss). I'd love to hear from you.

## Let's Begin!

Are you ready to build your first or next TypeScript-powered NPM package?

Let's start by setting up your project the right way in the next chapter.

# Project Configuration

Below are the required system and file configurations for creating NPM packages.

## 1. Setting Up Your System

Please ensure your system has the following installations:

- Git
- Node 22.13 (or greater)
- NPM 10.9 (or greater)

- Use the Git tutorial to install, update, or verify Git on your system.
- Use the package manager tutorial to install, update, or verify Node and NPM on your system.

## 2. Creating a Project Directory

Create a new folder for your project like so:

**Figure 1. Command Line**

```
1  mkdir thank-you-tweet-button-001
```

You may use any name. For example, this tutorial uses `thank-you-tweet-button-001`.

- Specify a name that is less than or equal to 214 characters.
- Use lowercase letters only in the name.
- Your package's name should not contain "js" or "node".

Afterward, navigate to your project directory using the command line.

**Figure 2. The Syntax to Change to a Different Directory**

```
cd path/to/thank-you-tweet-button-001
```

## 3. Creating a package.json File

Use NPM to initialize a package.json file for your project.

**Figure 3. Command Line**

```
npm init -y
```

Let's also configure Git.

## 4. Initializing a Git Repository

Create a `.git` repo in your project's root directory:

**Figure 4. Command Line**

```
git init
```

You now need to specify the files you want Git to ignore.

## 5. Specifying the Files Git Should Ignore

Create a `.gitignore` file in your project's root directory:

**Figure 5. Command Line**

```
touch .gitignore
```

Afterward, open the newly created `.gitignore` file and write the names of the files, folders, or file types you want Git to ignore.

**Here's an example:**

**Figure 6. .gitignore**

```
/node_modules
/dist
```

The snippet above instructs Git to ignore tracking the current directory's `node_modules` and `dist` folders.

- A current directory is the folder in which you are currently working.
- A current directory is sometimes called a "current working directory (CWD)" or "working directory."
- See the .gitignore file example to learn more about `gitignore`.

It's now time to stage and commit your recent changes.

## 6. Staging and Committing Your Project's Changes to Git

Enter the following command on your terminal to stage and commit your recent changes.

**Figure 7. Command Line**

```
git add -A && git commit -m "Initialize project using CodeSweetly's guide"
```

The command above tells Git to stage and commit all modified and untracked files in the project.

Let's now configure a remote repository for the project.

## 7. Configuring a GitHub Remote Repository

1. Go to the GitHub website and sign in or create an account if you do not have one.
2. After signing in, create a new GitHub repository. You may use `thank-you-tweet-button-001` or another name of your choice.

3. Once you've created a remote repository for your package, link your project's `.git` directory (located locally on your system) with the remote repository on GitHub. To connect to the remote repository, go to your package's root directory via your local terminal and run the `git remote add` command. Here's the syntax:

**Figure 8. Command Line**

```
git remote add origin https://github.com/your-username/remote-repo-name.git
```

- Replace `your-username` in the code above with your GitHub username. Likewise, replace `remote-repo-name` with your remote repository's name.
- See GitHub Docs to learn more about creating a GitHub repository.

You can also add the remote repo to your `package.json` file so that people who want to contribute to your project can easily access it.

**Figure 9. package.json: Add repository field (line 4-7)**

```
{
  "name": "thank-you-tweet-button-001",
  "version": "1.0.0",
  "repository": {
    "type": "git",
    "url": "https://github.com/your-username/app-repo-name.git"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  }
}
```

Let's provide the remote repo's Issues URL as the package's bug tracker. And an email people can use to report issues:

**Figure 10. package.json: Add bugs field (line 8-11)**

```
{
  "name": "thank-you-tweet-button-001",
  "version": "1.0.0",
  "repository": {
    "type": "git",
    "url": "https://github.com/your-username/app-repo-name.git"
  },
  "bugs": {
    "url": "https://github.com/your-username/app-repo-name/issues",
    "email": "your-project-email@host.com"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  }
}
```

Afterward, stage and commit your changes:

**Figure 11. Command Line**

```
git add -A && git commit -m "Configure project's remote repo"
```

Let's now push the local Git repository upstream.

## 8. Uploading Your Local Git Directory to the Remote Repo

After successfully connecting your local directory to the remote repository, you can begin pushing (uploading) your local project upstream. Here's how:

**Figure 12. Command Line**

```
git push -u origin main
```

The command above instructs Git to push the `.git` directory of your local `main` branch to the remote `origin` branch on GitHub.

Refreshing your remote repository's page should now reflect your upload.

The next step is to set up the project with TypeScript. So, let's do that now.

## 9. Installing TypeScript

Install TypeScript as a dev-dependency:

**Figure 13. Command Line**

```
npm i -D typescript@5.9.3
```

The command above tells NPM to install `typescript` as your package's dev-dependency.

- The `typescript` package is a library containing TypeScript's core functionality for type-checking and compiling your TypeScript and JavaScript code to the standard JavaScript version for any browser.
- `i` is the shorthand notation for `install`.
- The `-D` flag is the shorthand notation for `--save-dev`.
- We installed `typescript` as dev-dependencies because you only need it for your package's development and testing purposes. Users do not need it in production.

So, now that you've installed TypeScript, we can configure the TypeScript compiler.

## 10. Configuring the TypeScript Compiler

Developers use a `tsconfig.json` file to specify a project's root directory and the options the compiler needs to compile the project's files.

TypeScript treats the folder containing `tsconfig.json` as your project's root.

`tsconfig.json` is the default TypeScript configuration file name, but you may use custom names such as `tsconfig.custom.json` or `codesweetly-type-config.json`. This is useful in NPM package development when different configurations are needed for build, test, or IDE purposes.

Navigate to your package's root directory and create a root configuration file along with four custom configuration files:

**Figure 14. Command Line**

```
touch tsconfig.json tsconfig.base.json tsconfig.test.json tsconfig.cjs.json
↪  tsconfig.esm.json
```

- `tsconfig.base.json`: A base configuration file having universal compiler options that other TypeScript configuration files can inherit through the `extends` property.
- `tsconfig.json`: The project's root configuration file for IDE tools like VSCode to use to provide IntelliSense and error checking while you work.
- `tsconfig.test.json`: The configuration file containing the compiler options for test libraries, like Jest, to use during tests.
- `tsconfig.cjs.json`: Used to configure TypeScript to output CommonJS (CJS) Module compilations.
- `tsconfig.esm.json`: Used to configure TypeScript to output ECMAScript Module (ESM) compilations.

Open the newly created files and add the following configurations:

## `tsconfig.base.json`

**Figure 15. tsconfig.base.json**

```
{
  "compilerOptions": {
    "target": "es2018",
    "rewriteRelativeImportExtensions": true,
    "allowSyntheticDefaultImports": true,
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "noEmitOnError": true,
    "noImplicitReturns": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "removeComments": true,
    "skipLibCheck": true,
    "strict": true
  }
}
```

- `compilerOptions`: specifies how TypeScript should compile the project.
- `target` tells TypeScript the JavaScript version you want to compile your code into.
- `rewriteRelativeImportExtensions`: rewrites TypeScript file extensions such as `.ts`, `.tsx`, `.mts`, and `.cts` in relative `import` paths to their JavaScript equivalent extensions in output files.
- `allowSyntheticDefaultImports` tells TypeScript to allow you to declare default imports like `import feature from "package"` rather than `import * as feature from "package"` when the module you are importing does not have a default export.
- `esModuleInterop` tells TypeScript to make its compiled code interoperable (compatible) between CommonJS and ES Modules codebases by creating namespace objects for all imports.
- `forceConsistentCasingInFileNames` tells TypeScript to throw an error when a program tries to reference a file with a casing different from the file's name. For instance, referencing a `codesweetly.ts` file with `./CodeSweetly.ts` will throw an error.
- `noEmitOnError` specifies that TypeScript should not emit any compiled code if errors exist in the TypeScript project.
- `noImplicitReturns` tells TypeScript to issue an error if a function's code path does not return a value.
- `noUnusedLocals` tells TypeScript to issue an error if a codebase contains unused local variables.
- `noUnusedParameters` tells TypeScript to issue an error if a function has unused parameters.
- `removeComments` tells TypeScript to remove all comments when transpiling TypeScript files to JavaScript.
- `skipLibCheck` specifies that TypeScript should not type-check declaration (`.d.ts`) files.
- `strict` tells TypeScript to use "strict mode" to type-check your codebase.

## `tsconfig.json`

**Figure 16. tsconfig.json**

```
{
  "extends": "./tsconfig.base.json",
  "compilerOptions": {
    "module": "nodenext",
    "moduleResolution": "nodenext",
    "noEmit": true
  }
}
```

- `extends`: informs TypeScript to inherit the configurations in the `./tsconfig.base.json` file.
- `module`: specifies that TypeScript should use the file's extension or the `type` field in the nearest `package.json` file to determine the syntax of the compiled file (`.mts` or `modules` for ESM output and `.cts` or `commonjs` for CJS output).
- `moduleResolution`: tells TypeScript to use the latest Node.js module resolution strategy to resolve (locate) the modules you import into your TypeScript files.
- `noEmit`: tells TypeScript not to output any file. It should only type check and report errors.

## `tsconfig.test.json`

**Figure 17. tsconfig.test.json**

```
{
  "extends": "./tsconfig.base.json",
  "compilerOptions": {
    "module": "commonjs",
    "moduleResolution": "node10",
    "noEmit": true,
    "isolatedModules": true
  }
}
```

- `extends`: informs TypeScript to inherit the configurations in the `./tsconfig.base.json` file.

- `module`: specifies that TypeScript should use the CommonJS module syntax in the compiled file.
- `moduleResolution`: tells TypeScript to use the standard (loose) Node.js module resolution strategy of versions greater than 10 to resolve (locate) the modules you import into your TypeScript files. This lets the resolver find files automatically, even when the imports lack file extensions.
- `noEmit`: tells TypeScript not to output any file. It should only type check and report errors.
- `isolatedModules`: tells TypeScript to transpile each file at a time rather than perform cross-file analysis. This prevents program-wide module resolution conflicts by ensuring TypeScript just transpiles the file and leaves Jest to resolve the module import statements.

## `tsconfig.cjs.json`

**Figure 18. tsconfig.cjs.json**

```
{
  "extends": "./tsconfig.base.json",
  "compilerOptions": {
    "module": "commonjs",
    "moduleResolution": "node10",
    "outDir": "dist/cjs",
    "declaration": false
  }
}
```

- `extends`: informs TypeScript to inherit the configurations in the `./tsconfig.base.json` file.
- `module`: specifies that TypeScript should use the CommonJS module syntax in the compiled file.
- `moduleResolution`: tells TypeScript to use the standard (loose) Node.js module resolution strategy of versions greater than 10 to resolve (locate) the modules you import into your TypeScript files. This lets the resolver find files automatically, even when the imports lack file extensions.
- `outDir`: tells TypeScript to emit its CommonJS transpiled files into the `dist/cjs` folder.
- `declaration`: prevents TypeScript from generating `d.ts` type declaration files.

## tsconfig.esm.json

**Figure 19. tsconfig.esm.json**

```
{
  "extends": "./tsconfig.base.json",
  "compilerOptions": {
    "module": "es2020",
    "moduleResolution": "bundler",
    "outDir": "dist/esm",
    "declaration": true,
    "declarationDir": "dist/esm"
  }
}
```

- `extends`: informs TypeScript to inherit the configurations in the `./tsconfig.base.json` file.
- `module`: specifies that TypeScript should use the ES2020 module syntax in the compiled file.
- `moduleResolution`: tells TypeScript to use the bundler-compatible module resolution strategy to resolve (locate) the modules you import into your TypeScript files. The `bundler` mode is best for frontend libraries that bundlers like Vite, Webpack, and Rollup will consume.
- `outDir`: tells TypeScript to emit its ECMAScript Module (ESM) transpiled files into the `dist/esm` folder.
- `declaration`: tells TypeScript to generate `d.ts` type declaration files from your package's TypeScript and JavaScript files.
- `declarationDir`: tells TypeScript to emit its `d.ts` type declaration files into the `dist/esm` folder.

Stage and commit your changes:

**Figure 20. Command Line**

```
git add -A && git commit -m "Configure the typescript compiler"
```

After configuring the TypeScript compiler, continue to the next chapter to learn how to write a test case for your component.

# Testing Package's Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 1. Installing the Testing Tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 2. Specifying Jest as Your Project's Test Runner Tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 3. Configuring Jest to Test TypeScript Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 4. Configuring Jest to Compile TypeScript Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 5. Setting Up Jest's Testing Environment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 6. Creating Your Code Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 7. Writing Your Test Case

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 8. Developing the Package's Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 9. Running the Test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Commit Message Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Conventional Commits Message Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Enforcing the Conventional Commits Format

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Setting Up Husky

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Creating a Hook to Auto-Lint Commit Messages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Setting Up Commitlint GitHub Action

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 1. Create a Commitlint GitHub Action Workflow File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 2. Define the Commitlint GitHub Action Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 3. Test the Commitlint GitHub Action Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Compiling TypeScript to JavaScript

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## How to Compile Both ECMAScript and CommonJS Modules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Distinguishing Between Source Code and Distribution Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Specifying Items to Compile

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Defining Package's Entry Point

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Specifying Package's Declaration File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Local Testing of Unpublished Package

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Link-Install Your Package Globally in Your System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Creating a Directory for the Demo Website

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Creating a package.json File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Initializing a Git Repository

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Specifying the Files Git Should Ignore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Installing TypeScript

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Configuring TypeScript

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Installing Your Package from Your System's Global Folder to the Test App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Using the Link-Installed Package in the Test App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Creating the Demo Website's Homepage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Styling the Demo Website's Elements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Installing the Parcel Build Tool

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Specifying Parcel as Your Website's Development Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Unlinking Your Package from the Test App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Unlinking Your Package from the Global Folder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Creating README

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Creating LICENSE

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Publishing Package to NPM

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 1. Search Engine Optimization (SEO)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### Description

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### Keywords

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### Author

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### License

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### Homepage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 2. Specify the Files You Want to Publish to NPM

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 3. Confirm the Files NPM Will Publish

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 4. Confirm That Your Package Have Passing Tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 5. Compile Any Pending Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 6. Stage and Commit Any Recent Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 7. Push Your Local Git Directory to the Remote Repo

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 8. Sign In or Sign Up on the NPM Website

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 9. Log In to NPM via the Terminal

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 10. Confirm If Your Package's Name Is Available

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### 1. CLI command

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### 2. URL search

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### What to do if your chosen package name is currently in use by someone else

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 11. Publish Your Package!

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Local Testing of the Published Package



## 1. Install the Package



## 2. Import the Package



## 3. Run Your Local Server

# Production Testing of the Published Package

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 1. Stage and Commit Your Changes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 2. Set Up a GitHub Remote Repository for Your Demo Test App

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 3. Upload Your Local Git Directory to the Remote Repo

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 4. Sign In or Create an Account on the Vercel Website

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 5. Deploy Your Project to Vercel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 6. Test the Package on the Live Demo Website

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Updating Package's Versions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Example 1: Updating to a Patch Version

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Example 2: Updating to a Minor Version

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Example 3: Updating to a Major Version

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Automating Version Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 1. Create a Release GitHub Action Workflow File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 2. Define the Release GitHub Action Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 3. How to Overwrite semantic-release's Default Configurations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 4. Notify Developers That the Package Uses Automated Version Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# NPM Trusted Publishing Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Add a New Feature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Update an Existing Feature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Automating GitHub Releases

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Add the Preset Library to Your Release Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## Configure semantic-release to Automatically Publish the Package's Release Notes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Modularizing TypeScript Codebase

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 1. Identify Independent Elements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 2. Split the Elements You Want to Extract into Their Separate Modules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### `generateStarIcons.ts`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

### `showError.ts`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 3. Import the Extracted Elements into the `tweetButton.ts` File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 4. Configure Jest to Strip the Extension from Import Statements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## 5. Release the Latest Version of the Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

# Epilogue

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## What's Next?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.

## One Last Favor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/creating-npm-package-with-typescript.