

Fully Revised for 2026

Creating NPM Package

Your React TypeScript Guide to Create, Test, and Publish NPM Libraries

OLUWATOBI SOFELA

Creating NPM Package with React TypeScript

React TypeScript Guide to Create, Test, and Publish NPM Libraries

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



© 2023 - 2026 Oluwatobi Sofela

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without express written permission of the publisher. It is illegal to copy this book, post it to a website, or distribute it by any other means without permission, except for the use of brief quotations in a book review.

This book is provided “as is” and “as available” without any warranty. While the author has taken every precaution to ensure the accuracy of this book’s information, in no event shall the author, company, or publisher be liable in whatsoever way for any loss or damages, including but not limited to any consequential, indirect, special, or incidental damages caused or alleged to be caused directly or indirectly by this book’s content.

The names, contact information, characters, and incidents portrayed in this book are the work of the author’s imagination. Any resemblance to actual persons, events or localities is entirely coincidental.

First published in 2023

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 #reactjs.

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

#reactjs

Also By CodeSweetly

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

Contents

Introduction **1**
Welcome to NPM Package! 1
How This Book Can Help You 1
The Roadmap 2
Dependencies 2
Prerequisite 2
Questions and Comments 2
Let's Get It Started! 2

Project Configuration **4**
1. Set Up Your System 4
2. Create a Project Directory 4
3. Create a package.json File 5
4. Initialize a Git Repository 5
5. Specify the Files Git Should Ignore 5
6. Stage and Commit Your Project's Changes to Git 6
7. Configure a GitHub Remote Repository 6
8. Push Your Local Git Directory to the Remote Repo 8
9. Install React and TypeScript 8
10. Configure the TypeScript Compiler 11

Testing React Component **17**
1. Install the Testing Tools 17
2. Specify Jest as Your Project's Test Runner Tool 17
3. Configure Jest to Test TypeScript Code 17
4. Configure Jest to Compile TypeScript Files 17
5. Configure Jest's Testing Environment 17
6. Create Your Code Files 18
7. Write Your Test Case 18
8. Develop Your React Component 18

9. Run the Test 18

Commit Message Configuration **19**
Conventional Commits Message Syntax 19
Enforcing the Conventional Commits Format 19
Setting Up Husky 19
Creating a Hook to Auto-Lint Commit Messages 19

Setting Up Commitlint GitHub Action **20**
1. Create a Commitlint GitHub Action Workflow File 20
2. Define the Commitlint GitHub Action Workflow 20
3. Test the Commitlint GitHub Action Workflow 20

Styling React Components **21**
1. Create the Stylesheet 21
2. Apply CSS Styles to Your Component 21
3. Run the Test 21

Distinguishing Between Source Code and Distribution Code **22**

Specify Items to Compile **23**

Compiling React to JavaScript **24**
How to Compile Both ECMAScript and CommonJS Modules 24
How to Add the Project's Stylesheets to the `dist` Directory 24

Defining Package's Entry Point **25**

Specifying Package's Declaration File **26**

Local Testing of Unpublished Package **27**
1. Link-Install Your Package Globally in Your System 27
2. Create a NextJS Demo Website for Testing Your Package 27
3. Install Your Package from Your System's Global Folder to the Test-App 27
4. Configure Turbopack to Resolve Symlinks 27
5. Use the Link-Installed Package in Your Test-App 27
6. Unlink Your Package from the Test-App 28
7. Unlink Your Package from the Global Folder 28

Creating README **29**

Creating LICENSE **30**

Publishing Package to NPM **31**
1. Search Engine Optimization (SEO) 31
2. Confirm That Your Package Have Passing Tests 32
3. Compile Any Pending Code 32
4. Stage and Commit Your Project's Changes to Git 32
5. Specify the Files You Want to Publish to NPM 32
6. Confirm the Files NPM Will Publish 32
7. Stage and Commit Any Recent Changes 32
8. Push Your Local Git Directory to the Remote Repo 33
9. Sign In or Sign Up on the NPM Website 33
10. Log In to NPM via the Terminal 33
11. Confirm If Your Package's Name Is Available 33
12. Publish Your Package! 34

Local Testing of the Published Package **35**
1. Install the Package 35
2. Import the Package 35
3. Run Your Local Server 35

Production Testing of the Published Package **36**
1. Stage and Commit Your Changes 36
2. Set up a GitHub Remote Repository for Your Next.js Test App 36
3. Push Your Local Git Directory to the Remote Repo 36
4. Sign In or Create an Account on the Vercel Website 36
5. Deploy Your Project to Vercel 36
6. Test the Package on Your Live Demo Website 37

Updating Package's Versions **38**
Example 1: Updating to a Patch Version 38
Example 2: Updating to a Minor Version 38
Example 3: Updating to a Major Version 38

Automating Version Management **39**
1. Create a Release GitHub Action Workflow File 39
2. Define the Release GitHub Action Workflow 39
3. How to Overwrite semantic-release's Default Configurations 39
4. Notify Developers That the Package Uses Automated Version Management 39

NPM Trusted Publishing Configuration **40**

Add a New Feature 40
Update an Existing Feature 40

Automating GitHub Releases **41**
Add the Preset Library to Your Release Workflow 41
Configure semantic-release to Automatically Publish the Package's Release Notes 41

Modularizing TypeScript Codebase **42**
1. Identify Independent Elements 42
2. Split the Elements You Want to Extract into Their Separate Modules 42
3. Import the Extracted Elements into the `TweetButton.tsx` File . . . 42
4. Configure Jest to Strip the Extension from Import Statements . . . 43
5. Release the Latest Version of the Project 43

Epilogue **44**
What's Next? 44
One Last Favor 44

Introduction

Welcome to NPM Package!

NPM is a database (storage) for thousands of packages. Developers use it to share, manage, find, and install libraries.



Packages are sometimes called libraries, plugins, frameworks, or tools. It is a directory (or project) with a `package.json` file for recording information about it.

Publishing NPM packages provides an effective means to share code with others and enhance your development skills. It is an important step in contributing to the developer community and demonstrating a range of competencies, including:

- communication skills
- collaboration skills
- organizational skills
- technical skills
- documentation skills
- problem-solving skills

Becoming an NPM package publisher is a unique way to deliver a finished product that others can use as part of or as the main component of their apps.

This beginner-friendly book discusses everything you need to know to create, test, and publish NPM packages like a pro.

How This Book Can Help You

The best way to learn is through progressive practice. In other words, the easiest way to become a proficient package publisher is to proactively build NPM plugins—rather than merely reading or watching videos about them.

Accordingly, this book adopts a project-oriented approach to maximize hands-on practice while teaching the fundamental concepts of publishing NPM packages.

To get the most out of this book, open your text editor (like VS Code) and code along as you read. This way, you'll avoid getting stuck in "Tutorial Hell."

The Roadmap

The main project in this book is a Tweet Button, which you can make as simple or advanced as you prefer.

Don't let the project types scare you off! You are in for an exciting ride, where you will learn the fundamental concepts needed to publish any NPM package. So, give it a chance.

Dependencies

It is strongly recommended to use the exact dependency versions specified in this book. This practice minimizes the likelihood of encountering bugs introduced by newer versions.

After finishing this book's lessons, you can upgrade the dependencies to the latest version. If the bump (update) causes a bug, search online or the package's community for the appropriate fix.

Prerequisite

Knowing the basics of Git, GitHub, React, and TypeScript will make it easier to learn how to publish NPM packages. It's helpful to be familiar with these tools before you start.

Questions and Comments

If you have questions or comments about this book, email contact@codesweetly.com or send me a direct message on Twitter (@oluwatobiss).

Let's Get It Started!

The next chapter will show you how to set up the project.

Project Configuration

The following outlines the necessary system and file configurations for creating NPM packages.

1. Set Up Your System

Ensure your system includes the following installations:

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



- Refer to the Git tutorial to install, update, or verify Git on your system.
- See the package manager tutorial for steps to install, update, or verify Node and NPM.

2. Create a Project Directory

Create a new folder for your project as follows:

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`.



- Choose a name with 214 characters or fewer.
- Use only lowercase letters in your project name.
- Do not include “js” or “node” in your package name.

Next, 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. Create a package.json File

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

Figure 3. Command Line

```
npm init -y
```

You should also initialize Git at this stage.

4. Initialize a Git Repository

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

Figure 4. Command Line

```
git init
```

5. Specify 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.



- The current directory refers to the folder you are working in.
- A current directory is also known as the "current working directory (CWD)" or "working directory."

Now, stage and commit your recent changes.

6. Stage and Commit Your Project's Changes to Git

Run the following command in 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.

Next, configure a remote repository for the project.

7. Configure 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 on creating a GitHub repository.

You can also add the remote repo to your `package.json` file for easy access to people who want to contribute to your project.

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 "Create repository and bugs fields"
```

Next, push your local Git repository to the remote repository.

8. Push 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 to the remote repository. Here's how:

Figure 12. Command Line

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

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



If you refresh the main page of your remote repository, you should see your upload.

Now it's time to set up your project with React and TypeScript.

9. Install React and TypeScript

Install React and TypeScript as development dependencies:

Figure 13. Command Line

```
npm i -D react@19.2.4 typescript@6.0.3 @types/react@19.2.14
```

The command above tells NPM to install `react`, `typescript`, and `@types/react` as your package's development dependencies.

Note

- The `react` package is a library containing React's core functionality for creating React components.
- The `typescript` package is a library containing TypeScript's core functionality for type-checking and compiling your React code to JavaScript.
- The `@types/react` package contains React's type definitions, which TypeScript uses to type-check your React code.
- `i` is the shorthand notation for `install`.
- The `-D` flag is the shorthand notation for `--save-dev`.

We installed `typescript` and `@types/react` as dev-dependencies because you only need them for your package's development and testing purposes. Users do not need them in production.

You may wonder why React is installed as a development dependency if users need it in production. The following explains the reasoning.

Why Install React as a Package's Dev-Dependency?

Installing React as a development dependency prevents users from being required to download React when installing your package.

React requires only one copy of the `react` package per project. Therefore, specifying React as a dev-dependency prevents users from having to download React when installing your package. If their project already has a React copy, they wouldn't need to install another one.

One thing to note: your package will only work in projects that already have React installed. But there's a good workaround:

- Duplicate the `"react"` property from your package.json `"devDependencies"` field into the `"peerDependencies"` field as shown:

Figure 14. package.json: Add peerDependencies field (line 5-7)

```
{
  "devDependencies": {
    "react": "^19.2.4"
  },
  "peerDependencies": {
    "react": "^19.2.4"
  }
}
```

Specifying React as a peer dependency tells package managers to check whether the app installing your package includes the listed `peerDependencies`. If so, the application has the dependencies your package needs to work.

If the package manager cannot find the `peerDependencies`, some NPM versions (such as 7 and 11) will install them automatically. Other versions (such as 3-6) will display a warning that informs the user to install the dependencies manually.

dependencies vs. devDependencies vs. peerDependencies

- `"dependencies"` specify the packages a project depends on in production (when your app is running on a live server).
- The `"devDependencies"` field lists all the packages a project does not need in production, but requires for its local development and testing purposes.
- `"peerDependencies"` list all the packages a project expects its host application to have in its `node_modules` directory.

Stage and commit your changes:

Figure 15. Command Line

```
git add -A && git commit -m "Install react and typescript"
```

After installing React and TypeScript, proceed to configure the TypeScript compiler.

10. Configure 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 16. 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 17. tsconfig.base.json

```
{
  "compilerOptions": {
    "jsx": "react-jsx",
    "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.
- `jsx`: specifies how TypeScript should compile the JSX syntax in `.tsx` files to plain JavaScript. (The "react-jsx" value means TypeScript should change JSX syntaxes to `_jsx` calls that are optimized for production. This allows you to use JSX without explicitly including `import React from "react"` in every file.)
- `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 18. 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 19. tsconfig.test.json

```
{
  "extends": "./tsconfig.base.json",
  "compilerOptions": {
    "module": "nodenext",
    "moduleResolution": "nodenext",
    "noEmit": true,
    "isolatedModules": 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.
- `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 20. tsconfig.cjs.json

```
{
  "extends": "./tsconfig.base.json",
  "compilerOptions": {
    "module": "nodenext",
    "moduleResolution": "nodenext",
    "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 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.
- `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 21. 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 22. 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 React Component

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

1. Install the Testing Tools

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

2. Specify 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-react-typescript.

3. Configure 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-react-typescript.

4. Configure 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-react-typescript.

5. Configure 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-react-typescript.

6. Create 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-react-typescript.

7. Write 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-react-typescript.

8. Develop Your React Component

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

9. Run 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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-typescript.

Styling React Components

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

1. Create the Stylesheet

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

2. Apply CSS Styles to Your Component

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

3. Run 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-react-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-react-typescript.

Specify 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-react-typescript.

Compiling React 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-react-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-react-typescript.

How to Add the Project's Stylesheets to the `dist` Directory

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

`scripts/copy-css.js`

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

`scripts/create-package-json.js`

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

1. 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-react-typescript.

2. Create a NextJS Demo Website for Testing 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-react-typescript.

3. Install 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-react-typescript.

4. Configure Turbopack to Resolve Symlinks

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

5. Use the Link-Installed Package in Your 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-react-typescript.

6. Unlink 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-react-typescript.

7. Unlink 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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-typescript.

2. 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-react-typescript.

3. 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-react-typescript.

4. Stage and Commit Your Project's Changes to Git

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

5. 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-react-typescript.

6. 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-react-typescript.

7. 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-react-typescript.

8. 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-react-typescript.

9. 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-react-typescript.

10. 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-react-typescript.

11. 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-react-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-react-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-react-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-react-typescript.

12. 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-react-typescript.

Local 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-react-typescript.

1. Install the 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-react-typescript.

2. Import the 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-react-typescript.

3. Run Your Local 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-react-typescript.

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-react-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-react-typescript.

2. Set up a GitHub Remote Repository for Your Next.js 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-react-typescript.

3. 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-react-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-react-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-react-typescript.

6. Test the Package on Your 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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-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-react-typescript.

`TweetButtonPropsType.types.tsx`

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

3. Import the Extracted Elements into the `TweetButton.tsx` 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-react-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-react-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-react-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-react-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-react-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-react-typescript.