Пакеты и библиотеки

или как правильно использовать чужой код

Кормышев Егор ИСиП-301

14 марта 2024 г.

Термины и определения

NPM Пакет (или библиотека) - это набор переиспользуемумого кода опубликованный в перозитории

Репозиторий - облачное хранилище и система контроля версий для пакетов (сравнимо с github) Список пакетов NPM доступен по ссылке npmjs.com

Node Package Manager - менеджер пакетов Node.js предоставляющий такие функции для управления пакетами, как установка, удаление, обновление и другие

Начало работы с npm

Для начала нужно создать новый npm проект

Это можно сделать всего в 3 действия:

1. Создать пустую папку
2. В терминале/консоли перейти в эту папку
3. Выполнить команду `npm init`

Создание проекта I

С этого момента начинается создание проекта в интерактивном режиме NPM поможет вам составить конфигурацию проекта через ответы на вопросы

Список вопросов

1. `package name` - имя проекта (по умолчанию - имя папки)
2. `version` - версия проекта (по умолчанию - 1.0.0)
3. `description` - описание - краткое описание назначения и/или функционала проекта
4. `entry point` - главный файл проекта (по умолчанию - index.js)
5. `test command` - скрипт для тестированя (по умолчанию - просто вывод строки в консоль)
6. `git repository` – ссылка на git репозиторий (при наличии)

Создание проекта II

7. `keywords` - ключевые слова - "теги"по котором будут искать ваш пакет (например: framework, game, preprocessor)
8. `author` - автор - ваш ник на npm или github с почтой в особом формате (`coffeek-codes <kormyshev11@mail.ru>`)
9. `license` - Лицензия - самая популярная свободная лицензия MIT - подробнее здесь

Затем вы увидите вашу конфигурацию и последний вопрос с подтверждением.

Поздравляю! Вы создали свой первый npm проект!

Структура проекта и package.json I

Теперь после того, как вы создали свой проект или же какой-либо шаблон (например vite) в папке проекта вы увидите файл с названием package.json В проекте vite то,

что вы увидите в package.json можно поделить примерно следующие части:

1. Информация о проекте

```
"name": "vite",
"private": true,
"version": "0.0.0",
"type": "module",
```

Структура проекта и package.json II

2. Скрипты запуска проекта

```
"scripts": {
  "dev": "vite",
  "build": "tsc && vite build",
  "preview": "vite preview"
},
```

3. Зависимости

```
"devDependencies": {
  "typescript": "^5.2.2",
  "vite": "^5.1.6"
}
```

А также папку node_modules/

Структура проекта и package.json III

Эта папка содержит файлы **ВСЕХ ЗАВИСИМОСТЕЙ** в вашем проекте и у нее есть 1 главное правило

При передаче проекта с помощью контроля версий (git) или любыми другими способами

НИКОГДА НЕ ВКЛЮЧАЙТЕ ЭТУ ПАПКУ В ПРОЕКТ

Это может привести к потере и повреждению файлов зависимостей

Команды NPM

Основные команды NPM:

- `npm install <название пакета>` (кратко `npm i`) - установить пакет в проект
- `npm uninstall <название пакета>` (кратко `npm un`) - удалить пакет из проекта
- `npm run <название скрипта>` - запустить скрипт из раздела `scripts` файла `package.json`

- `npm install (i)` - установить все зависимости из файла `package.json`

ИМЕННО ТАК УСТАНАВЛИВАЮТСЯ ЗАВИСИМОСТИ ПРИ ПЕРЕДАЧЕ ПРОЕКТА

Первая зависимость |

Итак, установим первую зависимость

Для решения задач вам понадобится установить пакет `lodash-ts`
Сделать это мможно следующей командой:

```
npm install lodash-ts
```

Или более краткая форма:

```
npm i lodash-ts
```

Первая зависимость II

После этого в файле `package.json` появятся следующие строки:

После установки

```
"dependencies": {
  "lodash-ts": "^1.2.7"
}
```

Импорт и использование зависимостей в проекте

После установки вы можете использовать функции зависимостей в своем проекте
Для этого используется следующий синтаксис:

Пример Импорта

```
import <название функции> from <название пакета>
```

Пример для `lodash-ts`

lodash пример

```
import isArray from 'lodash-ts/isArray';
```