Домашняя работа №5: Основы npm (Node Package Manager)

Студент: Дмитриев Андрей Иванович
Группа: K3439
Вариант: Платформа для образовательных курсов и управления учебным процессом

---

1. Описание задачи

В рамках домашней работы №5 требовалось продемонстрировать знание основ работы с **npm** (Node Package Manager) — пакетным менеджером для JavaScript. Необходимо было:

- понимать структуру package.json;
- уметь устанавливать и управлять зависимостями;
- знать основные npm-команды;
- различать **production** и **development** зависимости;
- понимать **семантическое версионирование (SemVer)**;
- работать с **npm-скриптами**.

---

2. Что такое npm

2.1 Определение

npm (Node Package Manager) — это:

1. **Пакетный менеджер** для JavaScript/Node.js
2. **Реестр пакетов** (npmjs.com) — крупнейшая база open-source библиотек
3. **CLI-инструмент** для установки, удаления, обновления пакетов и запуска скриптов

2.2 Зачем нужен npm

Управление зависимостями

- автоматическая установка библиотек;
- контроль версий;

- разрешение конфликтов зависимостей.

Переиспользование кода

- доступ к готовым решениям;
- возможность публикации собственных пакетов.

Автоматизация

- запуск сборки, тестов, линтинга, деплоя через scripts.

Командная разработка

- одинаковые версии у всех участников;
- package-lock.json обеспечивает воспроизводимую установку.

3. Структура package.json

3.1 Основная информация

```
{
  "name": "front_it_school",
  "version": "0.1.0",
  "private": true,
  "description": "Платформа для управления образовательными курсами",
  "author": "IT School Team",
  "license": "MIT"
}
```

Назначение полей:

- name — имя пакета/проекта
- version — версия по SemVer
- private — защита от случайной публикации в npm
- description — описание
- author — автор/команда

- license — лицензия

3.2 Production зависимости (dependencies)


```
{
  "dependencies": {
    "@fullcalendar/daygrid": "^6.1.17",
    "@fullcalendar/interaction": "^6.1.17",
    "@fullcalendar/list": "^6.1.19",
    "@fullcalendar/react": "^6.1.17",
    "@fullcalendar/timegrid": "^6.1.17",
    "axios": "^1.9.0",
    "react": "^19.1.0",
    "react-dom": "^19.1.0",
    "react-router-dom": "^7.6.0",
    "react-scripts": "5.0.1",
    "react-toastify": "^11.0.5",
    "web-vitals": "^2.1.4"
  }
}
```


Production dependencies — библиотеки, которые нужны приложению для работы в production.

Ключевые пакеты проекта:

- **React**: react, react-dom — основа UI
- **Роутинг**: react-router-dom — навигация по страницам
- **HTTP-клиент**: axios — запросы к API
- **Календарь/расписание**: @fullcalendar/* — отображение расписания

- **Уведомления**: react-toastify — toast-сообщения
- **Метрики**: web-vitals — Core Web Vitals

Примечание: библиотеки тестирования обычно относят к devDependencies, но в CRA-проектах они часто встречаются в dependencies — это допустимо для учебного проекта.

3.3 Development зависимости (devDependencies)

```
{
  "devDependencies": {
    "@craco/craco": "^7.1.0",
    "css-minimizer-webpack-plugin": "^7.0.2"
  }
}
```

devDependencies — инструменты разработки/сборки, не требуются приложению “как продукту” в рантайме.

- @craco/craco — кастомизация CRA без eject
- css-minimizer-webpack-plugin — оптимизация/минификация CSS при сборке

3.4 npm-скрипты (scripts)

```
{
  "scripts": {
    "start": "craco start",
    "build": "craco build",
    "test": "craco test",
    "eject": "react-scripts eject"
  }
}
```

Назначение:

- npm start — запуск dev-сервера, HMR, отладка
- npm run build — production сборка (build/)
- npm test — запуск Jest тестов
- npm run eject — извлечение конфигов CRA (необратимо)

3.5 Browserslist

```
{
  "browserslist": {
    "production": [">0.2%", "not dead", "not op_mini all"],
    "development": ["last 1 chrome version", "last 1 firefox version", "last 1 safari version"]
  }
}
```

Browserslist определяет, под какие браузеры сборка должна адаптироваться:

- Babel транспилирует JS,
- Autoprefixer добавляет CSS-префиксы,
- оптимизируется размер бандла.

3.6 ESLint конфигурация

```
{
  "eslintConfig": {
    "extends": ["react-app", "react-app/jest"]
  }
}
```

ESLint проверяет качество кода и ошибки:

- react-app — набор правил CRA

- react-app/jest — правила для тестов

4. Семантическое версионирование (SemVer)

4.1 Формат

MAJOR.MINOR.PATCH

- **PATCH** — исправления багов без ломания API
- **MINOR** — новые фичи без ломания API
- **MAJOR** — breaking changes (несовместимые изменения)

Примеры:

- 1.0.0 → 1.0.1 — bugfix
- 1.0.1 → 1.1.0 — новая функциональность
- 1.1.0 → 2.0.0 — breaking API

4.2 Диапазоны версий в package.json

^ (caret) — разрешает MINOR/PATCH в рамках MAJOR:

"react": "^19.1.0"

Разрешено: 19.x.x, запрещено: 20.0.0

~ (tilde) — разрешает PATCH в рамках MINOR:

"react": "~19.1.0"

Разрешено: 19.1.x, запрещено: 19.2.0

Точная версия:

"react-scripts": "5.0.1"

Всегда ставится строго 5.0.1

5. Основные npm-команды

5.1 Установка

npm install

npm i

Установить пакет:

npm i axios

Установить как dev:

npm i -D @craco/craco

Установить конкретную версию:

npm i react@18.2.0

5.2 Удаление

npm uninstall axios

npm remove axios

npm rm axios

5.3 Обновление

Проверить устаревшие:

npm outdated

Обновить в рамках SemVer:

npm update

Поставить самую свежую:

npm i axios@latest

5.4 Скрипты

npm start

npm run build

npm test

npm run <script>

Список скриптов:

npm run

5.5 Безопасность

npm audit

npm audit fix

6. package-lock.json

package-lock.json фиксирует точные версии всех зависимостей и их дерево.

Зачем нужен:

- одинаковые версии у всех разработчиков;
- воспроизводимость сборки в CI/CD;
- контроль целостности (integrity).

Важно:

- ✅ коммитится в Git
- ❌ не редактируется вручную
- обновляется автоматически через npm

7. node_modules

node_modules/ — папка со всеми установленными пакетами:

- содержит прямые и транзитивные зависимости;
- не хранится в Git;
- восстанавливается командой npm install.

8. Практика в проекте IT School

8.1 Как npm используется в проекте

1. Установка окружения:

npm install

2. Запуск разработки:

npm start

3. Сборка production:

npm run build

4. Тестирование:

npm test

8.2 Примеры импортов

```
import axios from 'axios';
import { BrowserRouter } from 'react-router-dom';


import FullCalendar from '@fullcalendar/react';
import dayGridPlugin from '@fullcalendar/daygrid';
```

---

9. Лучшие практики

- регулярно проверять обновления: npm outdated
- следить за безопасностью: npm audit
- разделять dependencies и devDependencies
- не коммитить node_modules
- хранить package-lock.json в репозитории
- аккуратно относиться к npm audit fix --force

---

10. Выводы

В ходе выполнения домашней работы №5:

- изучены основы npm и назначение пакетного менеджера;
- разобрана структура package.json и роль его основных полей;
- освоены зависимости dependencies и devDependencies;
- изучены npm-скрипты (start, build, test);
- рассмотрено семантическое версионирование и диапазоны ^, ~, точные версии;
- изучена роль package-lock.json для воспроизводимой установки.

Дата выполнения: 21.01.2026