Презентацию подготовил:

Колесников Е. В.

5040102/50201

Node Package Manager

Что такое NPM?

- NPM (Node Package Manager) – стандартный менеджер пакетов для программной платформы Node.js.

  - Крупнейшее хранилище кода на одном языке в мире.

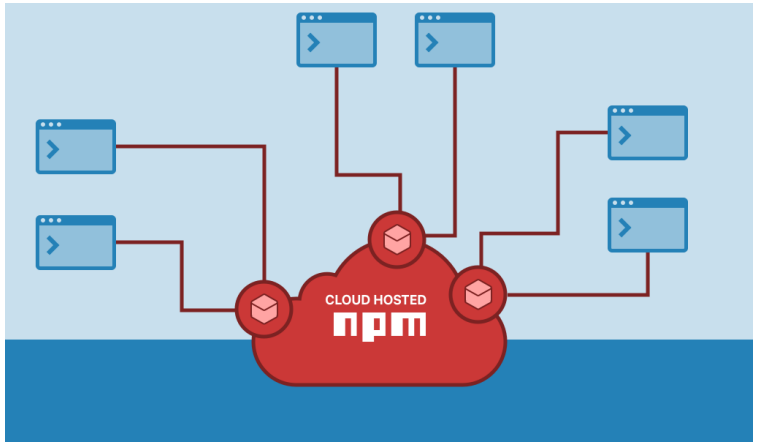
  - Изначально использовался для загрузки пакетов Node.js и управления их зависимостями.

  - Стал инструментом, применяемым и во фронтенд-разработке на JavaScript.

Для чего нужен NPM?

- Управление зависимостями (автоматическое отслеживание версий пакетов, их обновлений и зависимостями между ними).
- Установка, обновление и удаление библиотек и модулей.
- Запуск скриптов проекта (сборка, тесты).
- Публикация собственных пакетов.
- Создание и инициализация проектов с готовой структурой и файлов настроек.

Устройство NPM



- Состоит из трёх отдельных компонентов:

1) Веб-сайт npmjs.com для поиска пакетов, настройки профилей и управления разными аспектами работы с npm.
2) Интерфейс командной строки (запускается из терминала и используется большинством разработчиков).
3) Реестр пакетов JavaScript – большая общедоступная база данных программного обеспечения на языке JavaScript.

Структура npm-проекта

- package.json содержит список зависимостей, скрипты, информацию о проекте, npm всегда ориентируется на данный файл.
- Директория node_modules, в которой физически хранятся и в которую ставятся библиотеки.
- Package-lock.json фиксирует точные версии всех зависимостей, гарантируя одинаковую среду для всех разработчиков и серверов.

```
{
  "name": "my-project",
  "version": "1.0.0",
  "scripts": {
    "start": "node index.js",
    "test": "jest"
  },
  "dependencies": {
    "express": "^4.18.0"
  }
}
```

package.json

- Важно соблюдать требования к оформлению файла package.json, иначе npm не сможет прочитать содержимое.
- Разделы, определяющие метаданные проекта:

1) version – текущая версия программы
2) description – краткое описание возможностей
3) main – точка входа в приложение
4) private – возможность публикации
5) scripts – список скриптов для запуска
6) Depenencies – список зависимостей, без которых приложение не будет работать

```
"version": "1.0.0"
```

```
"description": "This utility
```

```
"main": "files/first.js"
```

```
"private": true
```

```
"scripts": {
```

```
"dependencies": {
    "express": "^4.17.1",
    "gm": "^1.22.0",
    "lodash": "^4.17.21",
    "music-metadata": "10.1.0"
  }
```

package.json

7) devDependencies – зависимости, нужные во время разработки.

8) engines – список версий Node.js, на которых приложение будет работать.

9) browserslist – список поддерживаемых браузеров и их версий.

10) author – информация об авторе проекта.

11) bugs – ссылка на баг-трекер проекта.

Основные команды NPM

Команда	Назначение
npm init (с параметром <-y> значения выставляются по умолчанию)	Запуск мастера создания package.json (название проекта, версия, описание).
npm install <name_packet>	Установка указанных в параметрах пакетов или всего, что указано в package.json.
npm uninstall <name_packet>	Удаление пакетов.
npm update	Обновление установленных библиотек

Основные команды NPM

Команда	Назначение
npm run <name_script>	Запуск скриптов из package.json.
npm list	Вывод библиотек и зависимостей, используемых в проекте.
npm publish	Загрузка библиотеки в публичный или приватный npm-реестр, делаю его доступным для установки другими пользователями.

Семантическое версионирование

- Для управления версиями пакетов NPM используются символы:

СИМВОЛ	ЗНАЧЕНИЕ	ПРИМЕР
~	Можно устанавливать только патчи пакета (1.2.x)	~1.2.0
^	Можно устанавливать только минорные версии и патчи (1.x.x)	^1.2.0
*	Последняя версия модуля	*
>	Выше указанной версии	>1.2.0
<	Ниже указанной версии	<1.2.0
=	Строго указанная версия модуля	=1.2.0
>=	Версия, равная указанной или выше	>=1.2.0
<=	Версия, равная указанной или ниже	<=1.2.0
-	Диапазон версий	1.2.0-1.3.4
latest	Последняя версия модуля	latest
||	Логическое ИЛИ	<1.2.1 || >1.3.4

Преимущества использования NPM

- Самый большой репозиторий JavaScript-пакетов
- Автоматическое управление зависимостями
- Удобный запуск скриптов через npm run
- Автоматическая установка вместе с Node.js
- Надежное версионирование
- Возможность легко публиковать собственные пакеты

Заключение

- С помощью NPM можно устанавливать библиотеки и модули, которые расширяют возможности языка программирования.
- NPM позволяет упростить управление зависимостями и ускорить разработку, обеспечивая доступ к крупнейшему репозиторию пакетов в мире.
- Освоение этого инструмента даёт разработчику много возможностей эффективного создания и поддержки проектов любой сложности.