What Is Dependency Management, and How Does It Work with Libraries Like React?

In software, a dependency is where one component or module in an application relies on another to function properly. Dependencies are common in software applications because it allows developers to use pre-built functions or tools created by others.

If you want to build out a React application, you will need to install the React dependencies. Without these dependencies, none of your code will work correctly and the application will display a list of errors.

When you are working with more complex projects, you will often need to rely on many dependencies. If a set of dependencies is not well managed or defined in a project, then that will lead to what is known as dependency hell.

To manage software dependencies in a project, you will need to use a package manager. A package manager is a tool used for installation, updates, and removal of dependencies. Many popular programming languages like JavaScript, Python, Ruby and Java, all use package managers.

In an earlier lesson, you were briefly introduced to one popular package manager called npm.

To create a new React project using Vite and npm you can run the following command:

```
npm create vite@latest my-react-app -- --template react
```

As you recall from the prior lessons, this will create all of the necessary boilerplate code needed to launch a new React application. Before you can launch the application, you will need to install the dependencies by running `npm install` or `npm i` for short.

You can view all of the dependencies in the `package.json` file which is located in the root directory of your project.

The `package.json` file is a key configuration file in projects that contains metadata about your project, including its name, version, and dependencies. It also defines scripts, licensing information, and other settings that help manage the project and its dependencies.

When you install dependencies, a `node_modules` folder will be added to your project.

The `node_modules` folder is where all the packages and libraries required by your project are stored. This folder contains the actual code for the dependencies listed in the `package.json` file, including both your project's direct dependencies and any dependencies of those dependencies.

The two core dependencies needed for a React project will be the `react` and `react-dom` packages:

```
"dependencies": {
  "react": "^18.3.1",
  "react-dom": "^18.3.1"
}
```

The `package.json` will list the current versions you are using for those packages. If you need to update any packages locally, you can run the `npm update` command. Or you can update all packages globally by running the `npm update -g` command.

In addition to the `package.json` file, you will also have a `package-lock.json` file. This file will lock down the exact versions of all packages that your project is using. When you update a package, then the new versions will be updated in the lock file as well.

Another important aspect of the `package.json` file are the dev dependencies:

```
"devDependencies": {
  "@eslint/js": "^9.17.0",
  "@types/react": "^18.3.18",
  "@types/react-dom": "^18.3.5",
  "@vitejs/plugin-react": "^4.3.4",
  "eslint": "^9.17.0",
  "eslint-plugin-react": "^7.37.2",
  "eslint-plugin-react-hooks": "^5.0.0",
  "eslint-plugin-react-refresh": "^0.4.16",
  "globals": "^15.14.0",
  "vite": "^6.0.5"
}
```

Dev dependencies are packages that are only used for development and not in production. An example of this would be a testing library like Jest. You would install Jest as a dev dependency because you only use it to test your project locally, and isn't needed for the application to run in production.

For the majority of this lesson, we have been focusing on npm. But there are other package managers like yarn and pnpm. So which package manager should you use for your project?

Well, the short answer is it depends.

If you are joining an existing project with a team, then all of those decisions will be made for you and you will use the existing package manager. If you are building out a project from scratch, then you will need to research the pros and cons of each manager and decide which one will better suit your needs.

Questions

What is a dependency?

A special type of testing software used in JavaScript and Python applications.

This is where multiple tasks are running simultaneously in the background.

This is where one component in an application relies on another to function properly.

A special type of compiler used to run JavaScript code in the browser.

What are dev dependencies?

Packages that are only used in production and not in development.

Packages that are only specific to testing libraries.

Packages that are only specific to animation libraries.

Packages that are only used for development and not in production.

Which of the following is NOT mentioned in the lesson as an example of a popular package manager for JavaScript?

`pip`

`npm`

```
yarn
```

```
pnpm
```

Navigated to What Is Dependency Management, and How Does It Work with Libraries Like React?