node -v 🡪 returns the version of node installed

npm -v 🡪 returns the version of npm

npm install create-react-app 🡪 installs create-react-app module in node modules

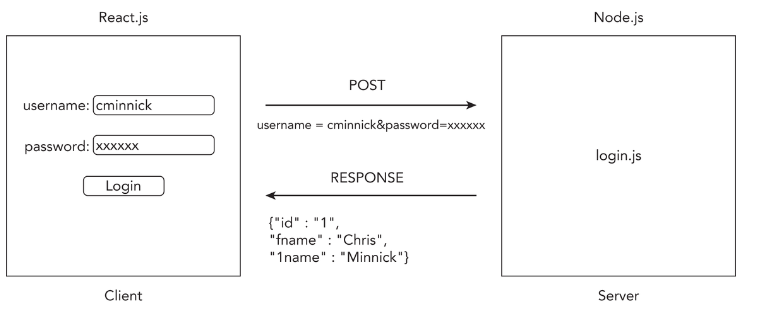
npx create-react-app my-firstapp OR 🡪 Creates a react app by name ‘my-FirstApp’

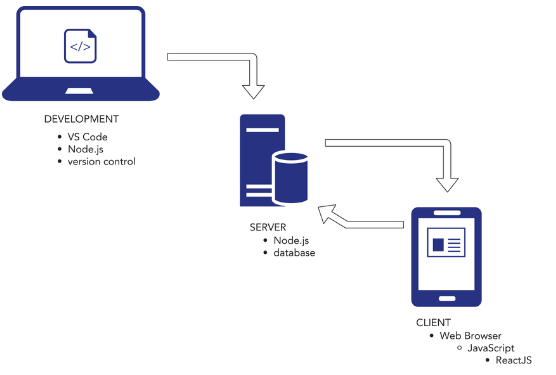
Open package.config and press Ctrl + S to resolve any build errors if you get know

npm install 🡪 It installs node modules folder from packages.json

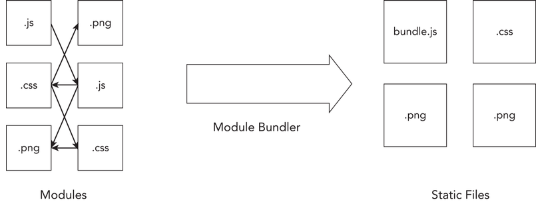
**Node JS**

* It started as a way to run Javascript on web servers so that same language can be used in both server side and client side





* Tasks done by Node.js
  + **Minification:** The process of removing white spaces, comments , line breaks making scripts, web pages and stylesheets more efficient and faster.
  + **Transpiling:** Not all browsers support the same version of javascript code. The process of converting one version of javascript code into another version is called as Transpiling. A screenshot of a computer

    Description automatically generated
  + **Module Bundling:** A typical website can make hundreds of scripts, stylesheets etc., If a browser were to download all of them it will slow down the page. Hence to overcome this we use bundlers. The main job of the bundler is to combine/bundle the javascript code so that it loads in the browser much faster. 
  + **Package Management:** As so many different programs are involved in JavaScript Development just installing, upgrading and keeping track of them can be quite complex. Hence we need a strong package manager.
  + **CSS Preprocessors:** A CSS preprocessor such as SCSS or LESS allows you to write style sheets that CSS lacks like variables, mathematical operations, functions, scope and nesting.
  + **Testing Frameworks:** Testing is an important part for any web project and the process of writing logic to test is a powerful tool.
  + **Build Automation:** If we a have complex manual process of writing code, testing, compiling, deploying then it’s better to automate where you can write a script or a program that can automate these things for us.

**NPM vs Yarn**

|  |  |  |
| --- | --- | --- |
| **Installation** | NPM comes pre-installed with Node.js  Ex: npm init 🡪 to initialize a project | Yarn needs to installed separately |
| **Speed** | Generally slower due to its sequential installation process. | Generally faster as it performs parallel installation of packages. |
| **Lock Files** | NPM uses package-lock.json for dependencies. | Yarn uses yarn.lock to lock dependencies. Yarn’s lock file ensures more consistency across different environments. |
| **Offline-Mode** | NPM doesn’t have built in offline capabilities. | Yarn supports offline-caching. Once package has been installed, yarn uses the cached-version without needing to re-download it. |
| **Deterministic Dependency Resolution:** | NPM 5 and above improved on this with package-lock.json | Provides deterministic dependency tree through yarn.lock ensuring same dependencies are installed every time. |
| **Command Syntax** | npm install , npm start | yarn add , yarn start |
| **Security** | NPM has audit command to fix vulnerabilities | Yarn also has security features and integrates with npm’s audit system. |
| **Workspaces** | Introduced workspaces in NPM 7, allowing for mono-repo support | Has workspace support |
| **Plug’n’Play** | NPM does not equivalent feature | Yarn 2 introduced Plug’n’Play feature that eliminates the need for having ‘node\_modules’ folder as it will directly reference the dependencies from cache. |
| **Community and Ecosystem** | NPM has larger user base with broader community | Yarn has huge community support but lesser than NPM. |

*Create React App*

* It installs a tool chain for react development and configures a boilerplate react application we can use as starting point for our application.

|  |  |
| --- | --- |
| * NPM is Node package manager manages packages and dependencies in the project | * npx is Node Package Executor. It’s main job is to execute the binaries present in node\_modules/bin folder |
| * Allows developers to publish their own packages to the npm registry. | * Npx executes binaries locally or from npm registry without the need to install them globally. |
| * Handles versioning and dependency resolution. | * Directly executes binaries without needing a script in package.json |
| npm install express // installs locally  npm install -g express // installs globally  npm run start // runs a script in package.json | npx create-react-app my-app // runs a package without installing  npx eslint . // runs locally installed package |

To create an app name below are the rules :

* It must be less than 214 characters long.
* The name can't start with a dot or underscore.
* The name can't have uppercase letters.
* It can't contain any characters that aren't allowed in URLs (such as ampersands and dollar signs) and that are “unsafe” in URLs (such as the percent symbol and spaces).

In addition to these rules, there are several common conventions for how Node.js packages, and therefore apps created using Create React App, are named:

* Keep it simple and as short as possible.
* Use only lowercase letters.
* Use dashes in place of spaces.
* Don't use the same name as a common Node.js package.