Setting up React Project

Lesson Plan

PW SKILLS

Topics Covered:

1. Setting up development environment for react.
2. Starter project with CDN.
3. Understanding npm and yarn and npx.

Setting up development environment for react

a. Install NodeJS:

- Go to the official Node.js website and download the installer for your operating system.
- Run the installer and follow the installation instructions to complete the setup.
- Verify that Node.js is installed by opening your terminal or command prompt and running the following command:

```javascript
npm -v
```

This command will show the latest version of Node.

b. Choose a Code Editor:

- Select a code editor that suits your preferences. Visual Studio Code (VS Code) is a popular choice and can be downloaded from the official website.
- Install the code editor by following the provided instructions.

Starter project with CDN

What is CDN?

- A CDN, or Content Delivery Network, is a distributed network of servers that work together to provide fast and reliable delivery of digital content, such as images, videos, web pages, and other static or dynamic assets, to end-users across the globe.

- The main purpose of a CDN is to reduce latency and improve the speed of content delivery by caching and distributing content across multiple geographic locations, so that users can access the content from a server closest to their location. This helps to minimize the distance that the content has to travel, reducing the time it takes to load the content and improving the user experience.

- CDNs also provide other benefits, such as improved scalability, increased reliability and availability, and better security, by offloading traffic from origin servers and protecting against DDoS attacks and other types of threats.

- Overall, a CDN can be an essential tool for organizations that need to deliver content quickly and efficiently to a global audience, such as e-commerce sites, media companies, and other content-heavy websites.

Why do we use CDN?

Organizations utilize CDNs for a variety of purposes, including:

- **Improved Performance:** Reduction of the distance that data must travel between the server and the user allows CDNs to contribute to an increase in website performance. The user experience can be enhanced by speeding up page loading times as a result of this.
- **High Availability:** By spreading a website's content over numerous servers in various places, CDNs can assist in enhancing a website's accessibility. If one of the servers fails, this can help to ensure that the website is still available.
- **Security:** In addition to standard security measures, CDNs can assist against distributed denial of service (DDoS) assaults on websites.
- **Traffic Management:** By dividing the load over several servers, CDNs can assist in managing traffic spikes. This may aid in preventing a website's slowdown or breakdown during periods of excessive traffic.
- **Cost Saving:** By unloading some of the traffic from the origin server, CDNs can assist in lowering the cost of distributing content. This can help firms save money by reducing the need for extra infrastructures like servers and bandwidth.

Where to get react CDN?

You can get React CDNs from its official website.

https://reactjs.org/docs/cdn-links.html

How to use CDN in the project file?

We can directly use CDN in our HTML file at the bottom of the body tag

```JavaScript
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <link rel="stylesheet" href="./index.css" />
    <title>React CDN</title>
  </head>
  <body>


<script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></sc
ript>
<script crossorigin
src="https://unpkg.com/react-dom@18/umd/react-dom.development.
js"></script>


  </body>
</html>
```

Why we do not use React CDN in production?

While it is possible to use React CDN (Content Delivery Network) in production, it is generally not recommended because of a few reasons:

1. **Security concerns:** Using an external CDN to host and deliver your React code means that you are relying on a third-party to serve your code, which can potentially introduce security vulnerabilities or risks.

2. **Performance issues:** CDNs can increase page load times if the user has to fetch the React code from the CDN. This can lead to a slower user experience, which can negatively impact your website or application.

3. **Control over dependencies:** Using a CDN means you have less control over which version of React is being served, and you may encounter compatibility issues with other dependencies in your application.

Instead, it is recommended to bundle and serve your React code along with your other assets from your own server or a cloud-based service like AWS S3 or Google Cloud Storage. This gives you more control over the security and performance of your application, and allows you to easily manage your dependencies.

Example:

Lets create a simple react app with the help of CDN Link

```
JavaScript
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Simple React App</title>
</head>
<body>
  <div id="root"></div>
```

This part is the basic HTML structure of the page. We have an empty div element with the **id** of **"root"** where the React component will be rendered.

```
JavaScript
<script crossorigin
src="https://unpkg.com/react@18/umd/react.development.js"></sc
ript>
<script crossorigin
src="https://unpkg.com/react-dom@18/umd/react-dom.development.
js"></script>
```

These two lines include the **CDN links** for **React** and **ReactDOM**. These are the core libraries required to work with React.

```javascript
  <script>
    // React component
    function App() {
      return React.createElement(
        'div',
        null,
        React.createElement('h1', null, 'Welcome to My React App'),
        React.createElement('p', null, 'This is a simple React application.')
      );
    }
```

This section defines the React component App using the React.createElement() method. Here, we're creating a functional component that returns the JSX (JavaScript XML){will be discussed in depth in further classes} structure using React.createElement(). The component consists of a div element containing an h1 element for the heading and a p element for the paragraph.

```javascript
    // Render the React component
    ReactDOM.render(
      React.createElement(App),
      document.getElementById('root')
    );
  </script>
</body>
</html>
```

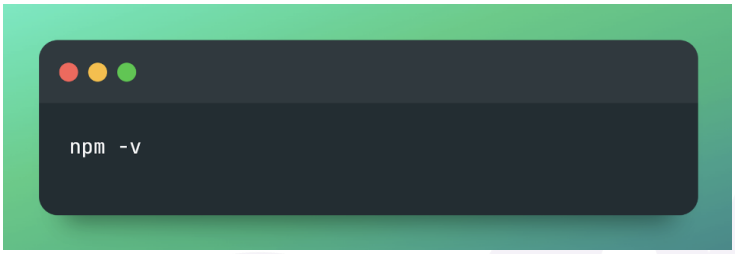
This part uses ReactDOM.render() to render the React component inside the HTML element with the id of "root". It takes two arguments: the first argument is the React component (App in this case), created using React.createElement(), and the second argument is the DOM element where the component will be rendered (document.getElementById('root')).

Understanding npm and yarn and npx

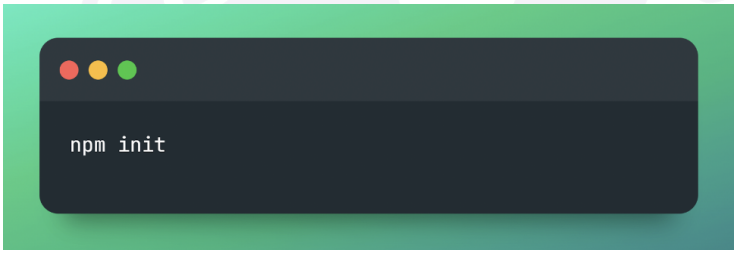
What is npm?

It is a **package manager** for the Node JavaScript platform**.npm** makes it easy for JavaScript developers to share, discover and reuse code, and manage dependencies in their projects. With npm, developers can install and use packages from the npm registry, as well as publish and distribute their own packages.
To use npm, you first need to install Node.js, which includes npm. **Node.js** is a JavaScript runtime built on Chrome's V8 JavaScript engine, which allows developers to run JavaScript on the server-side. Once Node.js is installed, you can use the npm command line interface (CLI) to install and manage packages.

The npm version that is currently running on your machine can be seen by using the command below:



package.json:
Go to the project's root directory and run the following command to generate the package.json file:

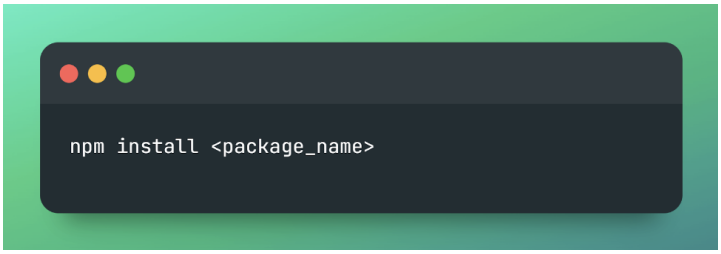


When we run the npm init command,it will gives you the following information:

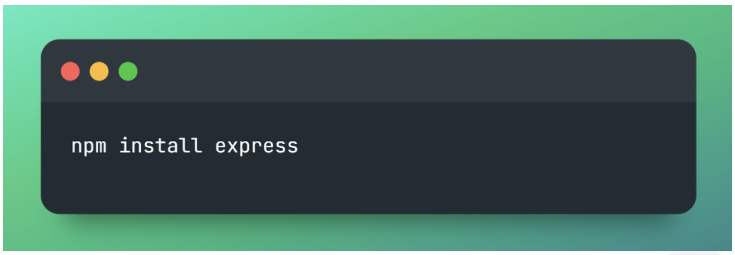
- Package name.
- Version.
- Test command
- Git repository.
- Keywords
- Author
- License

If we hit the enter or return button, It will accept default values.

Install a new Package:



For example if we want to install the express package;

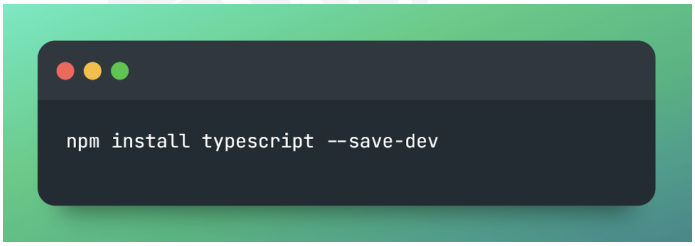


Install a package as a development dependency

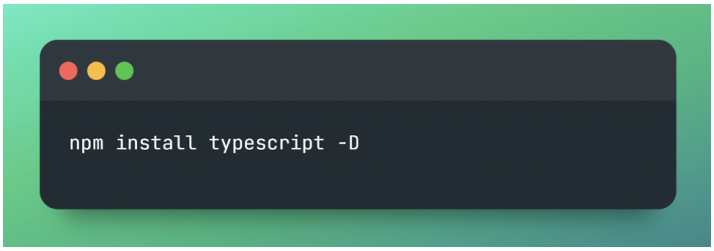
Sometimes, we want to install a package that runs only on the development environment.

We can install packages as devDependencies by using the **--save-dev or -D** flag when running the npm install command.

For example, to install the 'typescript' package as a devdependency, you would run the **following command:**



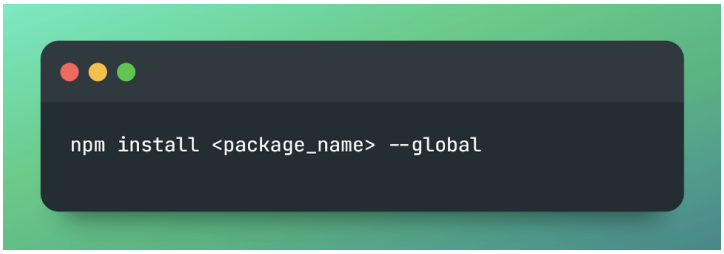
OR



These commands will add the typescript package to the **devDependencies** section of your package.json file, which specifies the packages that are only needed for development, such as testing and linting tools.

Install a package globally on your system

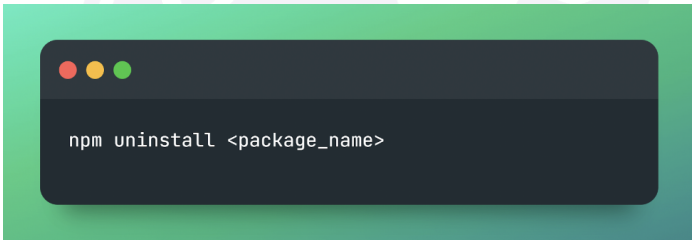
To install a package globally on your system, We use the following command



OR

Introduction to npm uninstall command

To remove a package from your current project, we use the npm uninstall command:



Why do we use NPM?

There are several reasons why npm is more useful tool in modern development:

1. **Dependency management:** You may manage your project's dependencies with the help of npm. A package is immediately added to your project's package.json file and saved in the node_modules directory when you use npm to install it. The packages that your project depends on are now simple to manage and update.

2. **Easy to use:** There are many packages available and it is simple to locate support when needed because npm is simple to use and has a big, active community.

3. **Cross-platform:** The fact that npm is the default package manager for Node.js implies that it is widely supported and can be used in a number of environments. npm runs on all major operating systems (Windows, macOS, and Linux).

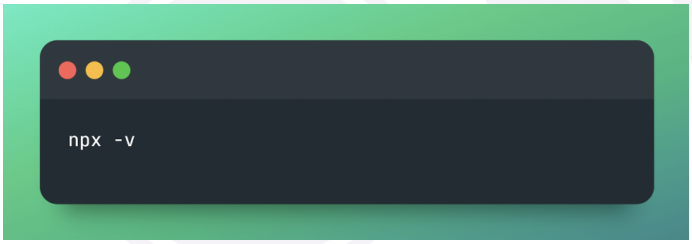
4. **Scripts:** You can define scripts in your package using npm. npm run command can be used to execute a json file. This may be helpful for constructing and testing your project.

What is npx?

npx is a command-line tool that comes bundled with Node.js. It allows you to execute Node.js packages directly without the need for manual installation or global installation. The term "npx" stands for **"Node Package Execute."**

When you run an npx command, it checks if the package you're trying to execute is installed locally in the current project. If it's not found, npx automatically downloads and installs the required package just for that execution, ensuring you have the latest version. Once the package is installed, npx runs the specified command from that package.

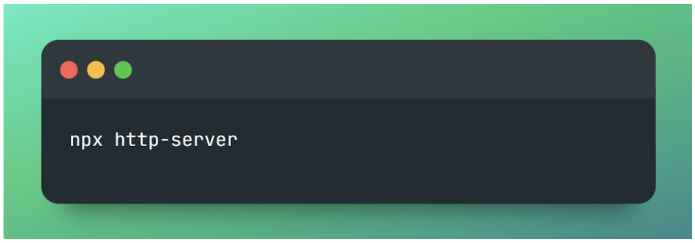
To check whether npx is installed or not, we use the following command:



Example:

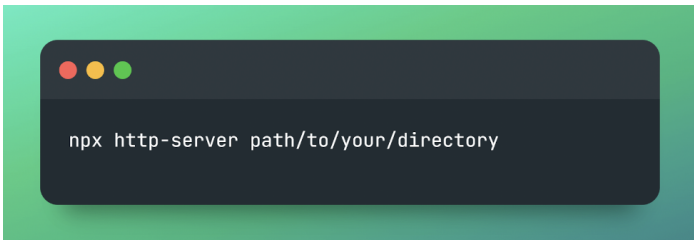
Let's say we want to use **`http-server`,** a simple command-line http server, to serve the files in your project. You don't want to install it globally on your system, but you want to use the latest version of the package.

You can use npx to run the http-server command without installing it globally by using the following command:



This will run http-server and serve the files in the current directory.

You can also specify a directory to serve by passing the directory path as an argument:



`npx` is a great tool that allows you to easily run commands from packages without having to install them globally and also allows you to run the latest version of a package without having to update your globally installed packages.

Npm vs npx

- If you're using a global package often, it would be helpful to install it with NPM so that executions can be faster (since it's already downloaded).
- But if you only want to use a global package one time (or rarely), and you don't want it downloaded to your device (and also occupy storage), you can use NPX to execute it directly.
- Sometimes we want to use some CLI tools but we don't want to install them globally so it is really helpful so that we can save some disk space and simply run them only when we need them

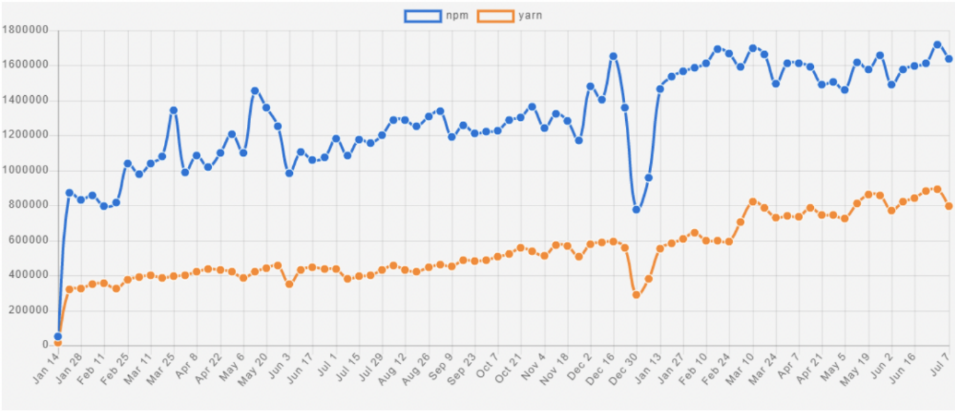
What is Yarn?

Yarn stands for **Yet Another Resource Negotiator**. Yarn is used to set up and maintain packages (such as libraries,frameworks, and tools) that you may use in your JavaScript projects, much like npm. You can use Yarn to install packages that are listed on the npm registry because it is compatible with the npm registry. In general, Yarn is a faster, more consistent replacement for npm.Depending on your preference, you can manage packages in your JavaScript applications using either npm or Yarn.
Yarn is another popular package manager for JavaScript. Yarn was intended to address some of the performance and security issues associated with using npm (at that time). Yarn was initially released by Facebook in 2016.

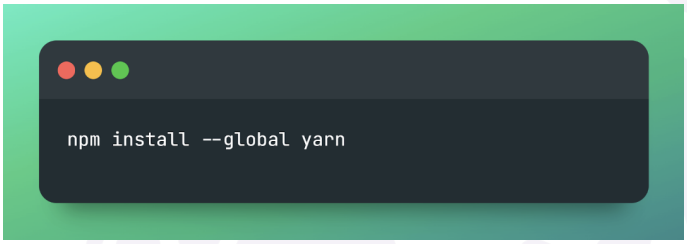
Yarn installs dependencies using the yarn command. It installs dependencies concurrently, i.e., in parallel, allowing you to add multiple files simultaneously. When you install dependencies, a lock file is created that stores the precise list of dependencies used for the project. This file is known as **yarn.lock.**

NPM vs Yarn

1. **Popularity:** In the previous five years, there have been more downloads of npm than of Yarn.



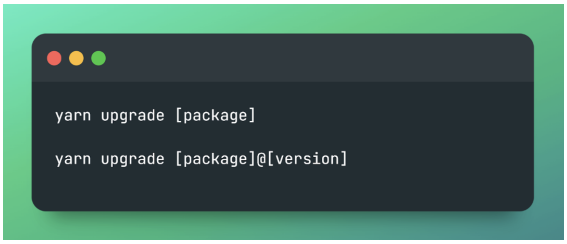
2. **Installation:** Yarn is available as an npm package. So, We can simply install it by running the following command on the terminal:



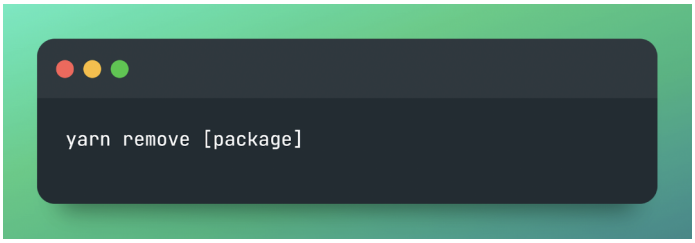
3. **Managing dependency:**

- Both Yarn and npm provide an auto generated lock file that has the entries of the exact versions of the dependencies used in the project.
- In npm it is called package-lock.json while in Yarn, it is called yarn.lock
- This file locks the dependencies to their stipulated versions during the installation process, after establishing the versioning parameters in the package.json file.
- When a dependency is installed, the lock file ensures that the same node modules file structure is maintained across all environments.Recently, Yarn released a new feature that makes the two package managers more aware of one another and makes it easier for developers to switch from npm to Yarn.
- Developers can import and install dependencies from the package-lock.json file of npm using this feature. It is a helpful upgrade, especially for anyone working in environments that mix yarn and npm or who want to switch their current projects to Yarn.
- To use this feature, just run the yarn import command in a repository having the package-lock.json file. As a result, Yarn will apply the resolution parameters in the package-lock.json file to generate a corresponding yarn.lock file.

4. Upgrading a dependency:



5. Removing dependency:



The package-lock.json file, created by NPM, is also supported by Yarn, making it easy to migrate version data from NPM to Yarn.

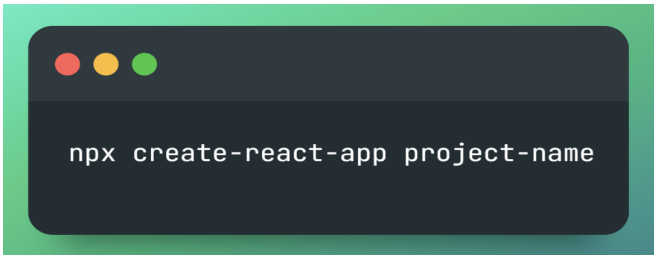
Npm	Yarn
It installs dependencies with the help of the `npm install` command.	t installs dependencies with the help of yarn add command.
It installs dependencies sequentially.	It installs dependencies in parallel.
The version lock file is known as package-lock.json.	The version lock file is known as yarn.lock.
It is slower when installing large files.	It is faster when installing large files.

What is create-react-app ?

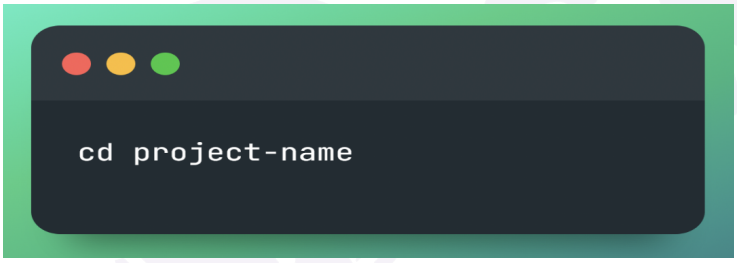
Create-React-App (CRA) is a tool developed by Facebook for creating and building React applications with minimal setup. It allows developers to quickly set up a new project with a basic file structure and a development server, without having to manually configure webpack or Babel. CRA also provides a set of scripts and tools that make it easy to test, build, and deploy React applications.

By using Create-React-App, developers can focus on writing code for their application instead of spending time configuring the development environment.

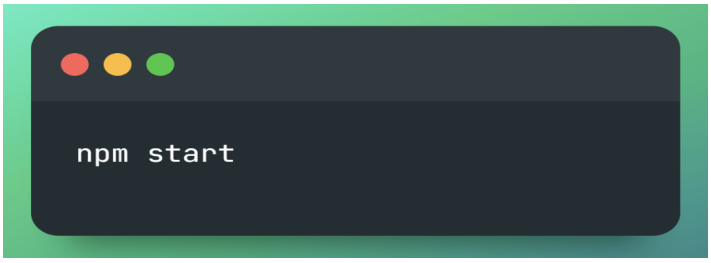
One of the methods listed below can be used to start a react project. Assume that node was installed.



Once the download is complete, move to the project file by using the following command .



Then start your project by using following commands:

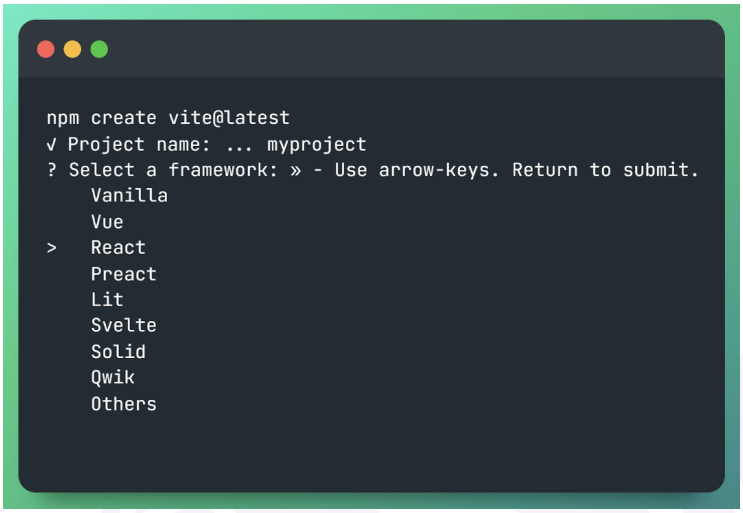


Note:
There is another way to create a react project by using vite . You can get all the details by visiting his official site.

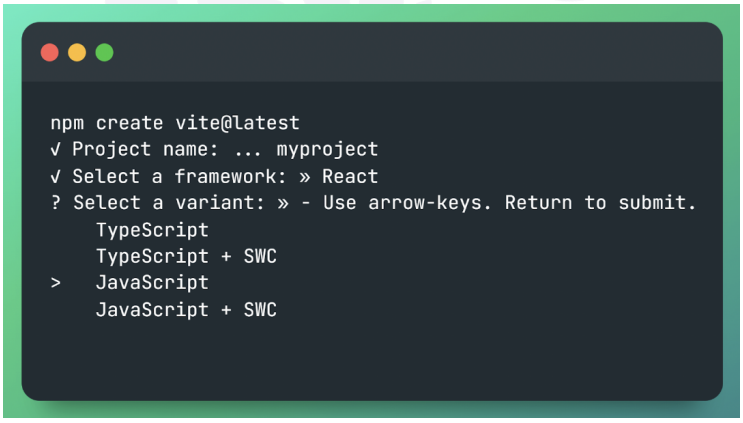
To create a vite project we use the following commands

```JavaScript
npm create vite@latest

or

yarn create vite
```

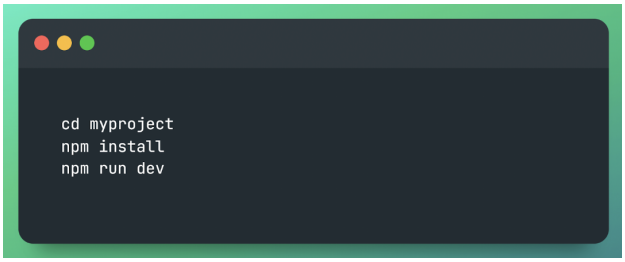
After that you select yes to continue, then it ask to give your project name.



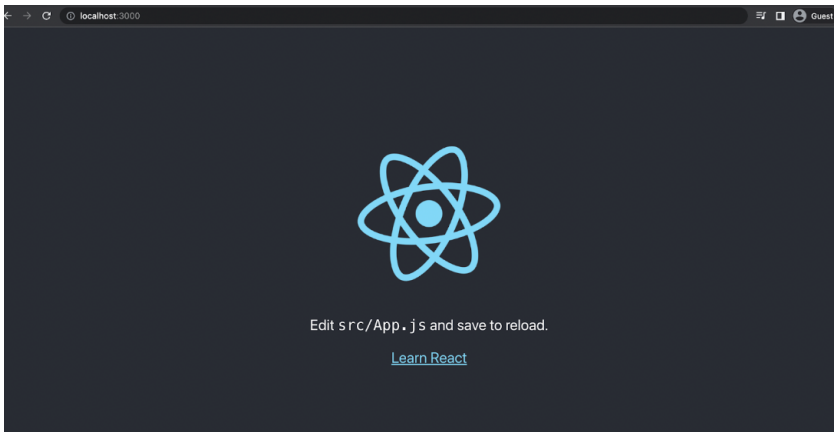
Then you select Javascript/Typescript . In this lecture we use Javascript



Then write the following command to start the project:.



Here We proceed with create-react-app...When we write `npm start`. Then localhost 3000 should host your React application. You can edit the content by adding text to the App.js file, and the browser will display the most recent changes. Press Ctrl + C in the CLI to terminate the server.



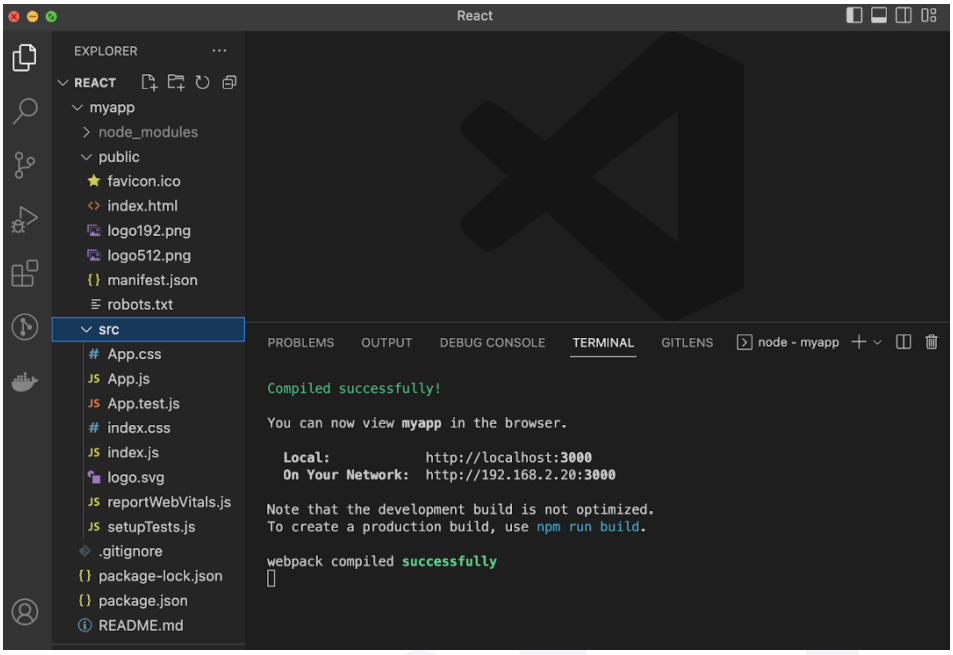
When We run the command **create-react-app <project-name>**, it creates a new directory with the given project name and generates a basic file structure for a React.js application. The generated file structure includes the following:

There are three folders in the following React boilerplate: node modules, public, and src. README.md, package.json,.gitignore, and yarn.lock are additional files. Some of you might have package-lock.json instead of yarn.lock.

- **node_modules:** Here We get all the necessary node packages of our application.

- **Public-**
  - **index.html:** Only one HTML file is there in the whole application.
  - **Manifest.json:** It is applied to transform the application into a progressive web app.
  - **favicon.ico:** It is an icon in the tab.

- **src-**
  - App.css, index.css - These are different CSS files.
  - index.js - A file which allows to connect all the components with index.html
  - App.js - A file where we usually import most of the presentational components
  - serviceWorker.js: is used to add progressive web app features
  - setupTests.js - to write testing cases
- **package.json-** It shows the list of packages the applications uses.
- **.gitignore-** React boilerplate comes with git initiated, and the .gitignore allows files and folders to be ignored when committing new changes.
- **README.md-** Markdown file to write documentation
- **yarn.lock or package-lock.json-** It means to lock the versions of the packages we have installed.



package.json vs package.lock.json

package.json:

The package.json file in a React project serves as a manifest file that contains metadata about the project, its dependencies, and other configurations. Here's an example package.json file for a React project:

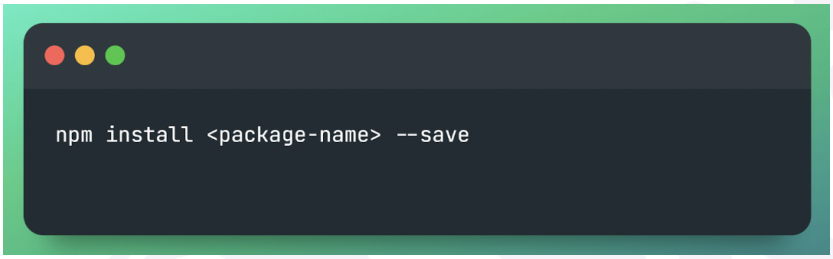
```
{
  "name": "myapp",
  "version": "1.0.0",
  "private": true,
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  },
  "devDependencies": {
    "react-scripts": "5.0.1"
  }
}
```

Properties of Package.json

- **name:** The name of your package.
- **description:** The purpose of your package.
- **scripts:** It contains a list of custom scripts which can be run with npm.
- **dependencies:** It contains dependencies which would be required by your package to work.
- **devDependencies:** It contains dependencies which are not required for your package to be used during production.

package.json holds important information of your package.

You'd also notice the **caret symbol (^)** used in front of the dependencies' version numbers. This symbol signifies to npm that higher major versions of this package can be installed.



The package would be saved in package.json's dependencies object with a value of '^'.

Let's say you pushed that project to a repository and cloned it on another system. When you run npm install on that other system, with the symbol there, npm would check if there is a higher major version, and if there is, that version is installed.

This feature comes with a good and a bad side effect.

Good: you always get to use the latest versions of packages.
Bad: the bad side is that latest versions may contain bugs or may contain incompatible codes for your codebase. npm released the **package-lock.json** file to solve this issue.

Package-lock.json

The package-lock.json file is automatically generated by npm when installing packages. It provides a detailed, deterministic record of the exact versions of packages and their dependencies, ensuring consistent installations across different environments. Here's a simplified example of a package-lock.json file for the above package.json:

```
{
  "name": "myapp",
  "version": "1.0.0",
  "lockfileVersion": 2,
  "requires": true,
  "dependencies": {
    "react": {
      "version": "18.2.0",
      "resolved": "https://registry.npmjs.org/react/-/react-18.2.0.tgz",
      "integrity": "sha512-..."
    },
    "react-dom": {
      "version": "18.2.0",
      "resolved": "https://registry.npmjs.org/react-dom/-/react-dom-18.2.0.tgz",
      "integrity": "sha512-..."
    }
  }
}
```

You'll find this file a bit similar to **package.json** - holding some information about your file, but there's more to it. Remember the caret symbol in package.json dependencies? You could manually remove that symbol when you install a dependency, but that would be stressful.

With package-lock.json, the versions of all dependencies your package would need are locked (-lock). This means when you run npm install on another system, npm checks this file to install the exact versions of the dependencies. If this file is absent, npm would then work with the caret symbol.

This means that you should also commit package-lock.json to your source control.

The major benefit of this file is that when your package is working perfectly (with all the dependencies used) on your system, you are sure that any attempt to install the package on another system would work exactly. It ensures similar dependencies in various environments where the package is being worked on.